COSC76/276 Artificial Intelligence
Fall 2022
Logical agents

Soroush Vosoughi
Computer Science
Dartmouth College
Soroush@Dartmouth.edu

Reminders

- SA5
- X-hours on Monday

Logic in general

- Logics are formal languages for representing information such that conclusions can be drawn from formal inference patterns

- Syntax defines the well-formed sentences in the language

- Semantics define the "meaning" or interpretation of sentences:
  - connect symbols to real events in the world
  - i.e., define truth of a sentence in a world

Logical inference

- The notion of entailment can be used for logic inference.

- $KB \mid\!-_i \alpha$ means *KB* derives a sentence α using inference procedure *i*

- *Sound* (or *truth preserving*):
  The algorithm **only** derives entailed sentences.
  i is sound iff whenever $KB \mid\!-_i \alpha$ *it is also true that* $KB \models \alpha$
- *Complete*:
  The algorithm can derive **every** entailed sentence.
  i is complete iff whenever $KB \models \alpha$ *it is also true that* $KB \mid\!-_i \alpha$

Validity and 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 false in all models

e.g., $A \wedge \neg A$

Example: College life

- Let's tell the knowledge base some sentences. Then ask if some other sentence is entailed.

1. If I did not eat dinner, that implies that either the fridge was empty, or an assignment was due and I procrastinated. If the fridge was empty, or an assignment was due, and I procrastinated, then I did not eat dinner.
2. If it is winter, and there was nice weather Sunday, I procrastinated.
3. If the fridge is empty, my housemate will be mad.

Write these sentences using propositional logic

Write sentences

Example: College life

- $\neg$ Dinner $\Leftrightarrow$ FridgeEmpty $\vee$ (AssignmentDue $\wedge$ Procrastinated)
- Winter $\wedge$ NiceWeatherSunday $\Rightarrow$ Procrastinated
- FridgeEmpty $\Rightarrow$ HousemateMad

- ask: if it is not winter, and I did not eat dinner, does that imply that my housemate is mad?
- KnowledgeBase $\wedge \neg$W $\wedge$ $\neg$D$\vDash$H?

Model checking

Method #1 for inference

– Seven symbols: D, F, P, A, W, H, N.
– Each symbol can take the value true or false.
– Consider all assignments of true/false values.

- If H is true for all models in which all sentences in (KB and ¬W¬D) are true, then H is entailed.

Conjunctive Normal Form (CNF)

- Boolean formulae are central to CS
  - Boolean logic is the way our discipline works
- Two canonical Boolean formulae representations:
  - CNF = Conjunctive Normal Form
    - A conjunct of disjuncts = (AND (OR …) (OR …))
    - "…" = a list of literals (= a variable or its negation)
    - CNF is used by Resolution Theorem Proving
- Can convert any Boolean formula to CNF

Review: Equivalence & Implication

- Equivalence is a conjoined double implication
  - $(X \Leftrightarrow Y) = [(X \Rightarrow Y) \wedge (Y \Rightarrow X)]$

Review: de Morgan's rules

- How to bring $\neg$ inside parentheses
  - (1) Negate everything inside the parentheses
  - (2) Change operators to "the other operator"

- $\neg(X \wedge Y \wedge \ldots \wedge Z) = (\neg X \vee \neg Y \vee \ldots \vee \neg Z)$

- $\neg(X \vee Y \vee \ldots \vee Z) = (\neg X \wedge \neg Y \wedge \ldots \wedge \neg Z)$

Review: Boolean Distributive Laws

- **Both** of these laws are valid:

- AND distributes over OR
  - $X \wedge (Y \vee Z) = (X \wedge Y) \vee (X \wedge Z)$
  - $(W \vee X) \wedge (Y \vee Z) = (W \wedge Y) \vee (X \wedge Y) \vee (W \wedge Z) \vee (X \wedge Z)$

- OR distributes over AND
  - $X \vee (Y \wedge Z) = (X \vee Y) \wedge (X \vee Z)$
  - $(W \wedge X) \vee (Y \wedge Z) = (W \vee Y) \wedge (X \vee Y) \wedge (W \vee Z) \wedge (X \vee Z)$

Logical equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are logically equivalent iff they are true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

These are important to know

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

CNF rules

1. Eliminate $\Leftrightarrow$, replace with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
2. Eliminate $\Rightarrow$, replace with $\neg\alpha \vee \beta$
3. Move $\neg$ inwards:
   - $\neg(\neg\alpha)$:α
   - $\neg(\alpha \wedge \beta)$:$\neg\alpha \vee \neg\beta$ (DeMorgan)
   - $\neg(\alpha \vee \beta)$:$\neg\alpha \wedge \neg\beta$ (DeMorgan)
4. Distribute $\vee$ over $\wedge$:
   - $\alpha \vee (\beta \wedge \gamma) \rightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Now we have a conjunction of disjunctions of literals:

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

Example: College life

- Let's tell the knowledge base some sentences. Then ask if some other sentence is entailed.

1. If I did not eat dinner, that implies that either the fridge was empty, or an assignment was due and I procrastinated. If the fridge was empty, or an assignment was due, and I procrastinated, then I did not eat dinner.
2. If it is winter, and there was nice weather Sunday, I procrastinated.
3. If the fridge is empty, my housemate will be mad.

Write these sentences using propositional logic

Write sentences

CNF example

- College life
  - $\neg D \Leftrightarrow F \vee (A \wedge P)$
  - $W \wedge N \Rightarrow P$
  - $F \Rightarrow H$

1. Eliminate $\Leftrightarrow$, replace with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
2. Eliminate $\Rightarrow$, replace with $\neg\alpha \vee \beta$
3. Move $\neg$ inwards:
   - $\neg(\neg\alpha)$:α
   - $\neg(\alpha \wedge \beta)$:$\neg\alpha \vee \neg\beta$ (DeMorgan)
   - $\neg(\alpha \vee \beta)$:$\neg\alpha \wedge \neg\beta$ (DeMorgan)
4. Distribute $\vee$ over $\wedge$:
   - $\alpha \vee (\beta \wedge \gamma) \rightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Write together

CNF example

1. Eliminate double implications
   - $\neg D \Leftrightarrow F \vee (A \wedge P)$
     - $\neg D \Rightarrow F \vee (A \wedge P)$
     - $F \vee (A \wedge P) \Rightarrow \neg D$
   - $W \wedge N \Rightarrow P$
   - $F \Rightarrow H$

CNF example

2. Replace implications with $\neg\alpha \vee \beta$:

- $\neg D \Rightarrow F \vee (A \wedge P)$
  - $\neg(\neg D) \vee (F \vee (A \wedge P))$
- $F \vee (A \wedge P) \Rightarrow \neg D$
  - $\neg(F \vee (A \wedge P)) \vee \neg D$
- $W \wedge N \Rightarrow P$
  - $\neg(W \wedge N) \vee P$
- $F \Rightarrow H$
  - $\neg F \vee H$

CNF example

3. Move ¬ inwards:
   - $\neg(\neg D) \lor (F \lor (A \land P))$
     - $D \lor (F \lor (A \land P))$
   - $\neg(F \lor (A \land P)) \lor \neg D$
     - $(\neg F \land \neg(A \land P)) \lor \neg D$
   - $(\neg F \land (\neg A \lor \neg P)) \lor \neg D$
     - $\neg(W \land N) \lor P$
   - $\neg W \lor \neg N \lor P$
     - $\neg F \lor H$

CNF example

4. Distribute ∨ over ∧:
   - $D\vee(F\vee(A\wedge P))$
   - $(\neg F\wedge(\neg A\vee\neg P))\vee\neg D$
   - $\neg W\vee\neg N\vee P$
   - $\neg F\vee H$

- After:
   - $D\vee((F\vee A)\wedge(F\vee P))$
     - $(D\vee F\vee A)\wedge(D\vee F\vee P))$
   - …

CNF example

- It's in CNF! A conjunction of disjunctions!
  - $(D \vee F \vee A) \wedge (D \vee F \vee P))$
  - $(\neg F \vee \neg D) \wedge (\neg A \vee \neg P \vee \neg D))$
  - $\neg W \vee \neg N \vee P$
  - $\neg F \vee H$

CNF example

- Sanity check. Is
- $\neg D \Leftrightarrow F \lor (A \land P) \equiv (D \lor F \lor A) \land (D \lor F \lor P))$?
- Let's assume $\neg D$. Then $F \lor (A \land P)$ is true, from the double implication. Now looking at the CNF form:
- $(\text{False} \lor F \lor A) \land (\text{False} \lor F \lor P))$
- $(F \lor A) \land (F \lor P))$
- So either F is true, or both A and P are true, which is the same result we got from the original form. Sane.

Summary

- Methods for inference:
  - Model checking
  - Proofs, which need the sentences in CNF
- CNF
  - Conjunction of disjunctions
  - Any propositional logic sentence can be written in CNF using the logical equivalences

Look Back

- Goal inference:
  - Derive new sentences from old ones
  - Tell -> KB -> Ask

- Validity: a sentence is valid if it is true in all models
- Satisfiability: a sentence is satisfiable if it is true in some model

- Model checking enumeration

- Conjunctive Normal Form
- Logical equivalence

Today's learning objectives

- Algorithms for inference

The deduction theorem

For any sentences α,β,

$$\alpha \models \beta \text{ iff } (\alpha \Rightarrow \beta) \text{ is valid.}$$

- Entailment is a concept, and involves two sentences.
- $\alpha \Rightarrow \beta$ is a single sentence in PL.

How do we prove that $\alpha \Rightarrow \beta$ is valid?

Model checking? What would happen if we added hundreds of symbols, none of which appear in the KB or in α or β? Model checking would look silly

Proof by contradiction

Goal: prove that $\alpha \Rightarrow \beta$ is valid.

Maybe easy to prove a sentence unsatisfiable:

$A \wedge \neg A$ is unsatisfiable, even if we add new sentences, due to monotonicity.

So prove $\neg(\alpha \Rightarrow \beta)$ is unsatisfiable. In CNF:

- $\neg(\neg\alpha \vee \beta)$ unsatisfiable
- $\alpha \wedge \neg\beta$ unsatisfiable

So, add $\neg\beta$ to KB and show there are no models for which this is true. (Proof by contradiction.)

Inference rules: modus ponens

- If in the KB there are $\alpha \Rightarrow \beta$ and α, then the sentence β can be inferred

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

Inference rules: AND elimination

- If in the KB there are $\alpha \wedge \beta$, then any of the conjuncts can be inferred

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

Inference rules

- All logical equivalences can be applied as inference rules
  - Example:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Resolution rule

- One specific resolution rule will be the base of our procedure for inference

$$\frac{l_1 \vee l_2 \ldots \vee l_k, \quad m_1 \vee \ldots m_n}{l_1 \ldots l_{i-1} \vee l_{i+1} \vee \ldots l_k \vee m_1 \ldots m_{j-1} \vee m_{j+1} \vee \ldots m_n},$$

where l_i and m_j are complementary literals

Resolution rule example

- Does it make sense? Consider this case:
- $(A \lor B \lor C \lor D) \land (\neg A \lor E \lor F)$.
- A true: we can see from $(\neg A \lor E \lor F)$ that either E is true or F is true.
- A false: at least one of B, C, or D must be true.
- Add $B \lor C \lor D \lor E \lor F$ to KB

Does the resolution rule work?

Resolution Examples

$(A \vee B \vee C)$

$(\neg A)$

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

$(A \vee B \vee C)$

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

"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."

$(A \vee B)$

$(\neg A \vee B)$

"If A or B is true, and not A or B is true, then B must be true."

$\leftarrow$ Factoring

Resolution Examples

$(A \vee B \vee C)$

$(\neg A)$

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

$\overline{}$

$\therefore (B \vee C)$

$(A \vee B \vee C)$

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

"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."

$\overline{}$

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

$(A \vee B)$

$(\neg A \vee B)$

"If A or B is true, and not A or B is true, then B must be true."

$\overline{}$

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

More Resolution Examples

- (P Q ¬R S) with (P ¬Q W X) yields
  - Order of literals within clauses does not matter.
- (P Q ¬R S) with (¬P) yields
- (¬R) with (R) yields
- (P Q ¬R S) with (P R ¬S W X) yields
- (P ¬Q R ¬S) with (P ¬Q R ¬S) yields
- (P ¬Q ¬S W) with (P R ¬S X) yields
- ((¬ A) (¬ B) (¬ C) (¬ D)) with ((¬ C) D) yields
- ((¬ A) (¬ B) (¬ C)) with ((¬ A) C) yields
- ((¬ A) (¬ B)) with (B) yields
- (A C) with (A (¬ C)) yields
- (¬ A) with (A) yields

More Resolution Examples

- (P Q ¬R S) with (P ¬Q W X) yields <u>(P ¬R S W X)</u>
  - <u>Order of literals within clauses does not matter.</u>
- (P Q ¬R S) with (¬P) yields <u>(Q ¬R S)</u>
- (¬R) with (R) yields <u>( ) or FALSE</u>
- (P Q ¬R S) with (P R ¬S W X) yields <u>(P Q ¬R R W X) or (P Q S ¬S W X) or TRUE</u>
- (P ¬Q R ¬S) with (P ¬Q R ¬S) yields <u>None possible</u>
- (P ¬Q ¬S W) with (P R ¬S X) yields <u>None possible</u>
- ((¬ A) (¬ B) (¬ C) (¬ D)) with ((¬ C) D) yields <u>((¬ A) (¬ B) (¬ C))</u>
- ((¬ A) (¬ B) (¬ C)) with ((¬ A) C) yields <u>((¬ A) (¬ B))</u>
- ((¬ A) (¬ B)) with (B) yields (¬ A)
- (A C) with (A (¬ C)) yields <u>(A)</u>
- (¬ A) with (A) yields <u>( ) or FALSE</u>

Resolution rule

```
(OR   A   B   C   D)
(OR  ¬A  ¬B   F   G)
----------------------------
(OR  C  D  F  G)
```

```
(OR   A   B   C   D)
(OR  ¬A  ¬B  ¬C  )
----------------------------
(OR  D)
```

Are these correct?

Resolution rule typical error

The resolution rule resolves only ONE Literal Pair.
If more than one pair, result always = TRUE.

(OR A B C D)
(OR ¬A ¬B F G)

(OR C D F G)

This is wrong

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR D)

This is wrong

(OR A B C D)
(OR ¬A ¬B F G)

(OR B ¬B C D F G)

This is correct and simplifies to TRUE

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR A ¬A B ¬B D)

This is correct and simplifies to TRUE

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 $KB \wedge \neg\alpha$ and hence we **cannot** entail the query.

Inference by Resolution

- KB is represented in CNF
  - KB = AND of all the sentences in KB
  - KB sentence = clause = OR of literals
  - Literal = propositional symbol or its negation

- Find two clauses in KB, one of which contains a literal and the other its negation
  - Cancel the literal and its negation
  - Bundle everything else into a new clause
  - Add the new clause to KB
  - Repeat

Pseudo-code for resolution

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: 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 pair of clauses C_i, C_j in clauses do
        resolvents ← PL-RESOLVE(C_i, C_j)
        if resolvents contains the empty clause then return true
        new ← new ∪ resolvents
    if new ⊆ clauses then return false
    clauses ← clauses ∪ new
```

Proof by contradiction, i.e., show KB ^ ~α: unsatisfiable

Resolution example

Stated in Propositional Logic

- "Laws of Physics" in the Wumpus World:
  - "A breeze in B11 is equivalent to a pit in P12 or a pit in P21."

    $(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}))$
- Particular facts about a specific instance:
  - "There is no breeze in B11."

    $(\neg B_{1,1})$
- Goal or query sentence:
  - "Is it true that P12 does not have a pit?"

    $(\neg P_{1,2})$

Example: Conversion to CNF

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

1. Eliminate ⇔, replace with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
2. Eliminate ⇒, replace with $\neg\alpha \vee \beta$
3. Move ¬ inwards:
   - $\neg(\neg\alpha):\alpha$
   - $\neg(\alpha \wedge \beta):\neg\alpha \vee \neg\beta$ (DeMorgan)
   - $\neg(\alpha \vee \beta):\neg\alpha \wedge \neg\beta$ (DeMorgan)
4. Distribute ∨ over ∧:
   - $\alpha \vee (\beta \wedge \gamma) \rightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Write together

Example: Conversion to CNF

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

1. Eliminate $\Leftrightarrow$ by 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 $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$ and simplify.

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

3. Move $\neg$ inwards using de Morgan's rules and simplify.

   $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$, $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$

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

4. Apply distributive law ($\wedge$ over $\vee$) and simplify.

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

Example: Conversion to CNF

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

From the previous slide we had:

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

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

KB =

Often, Won't Write "$\vee$" or "$\wedge$"

(we know they are there)

...

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

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

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

(same)

...

Resolution example

Resulting Knowledge Base stated in CNF

- "Laws of Physics" in the Wumpus World:

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

- Particular facts about a specific instance:

  $(\neg B_{1,1})$

- Negated goal or query sentence:

  $(P_{1,2})$

Resolution example

A Resolution proof ending in ( )

- Knowledge Base at start of proof:

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

A resolution proof ending in ( ):

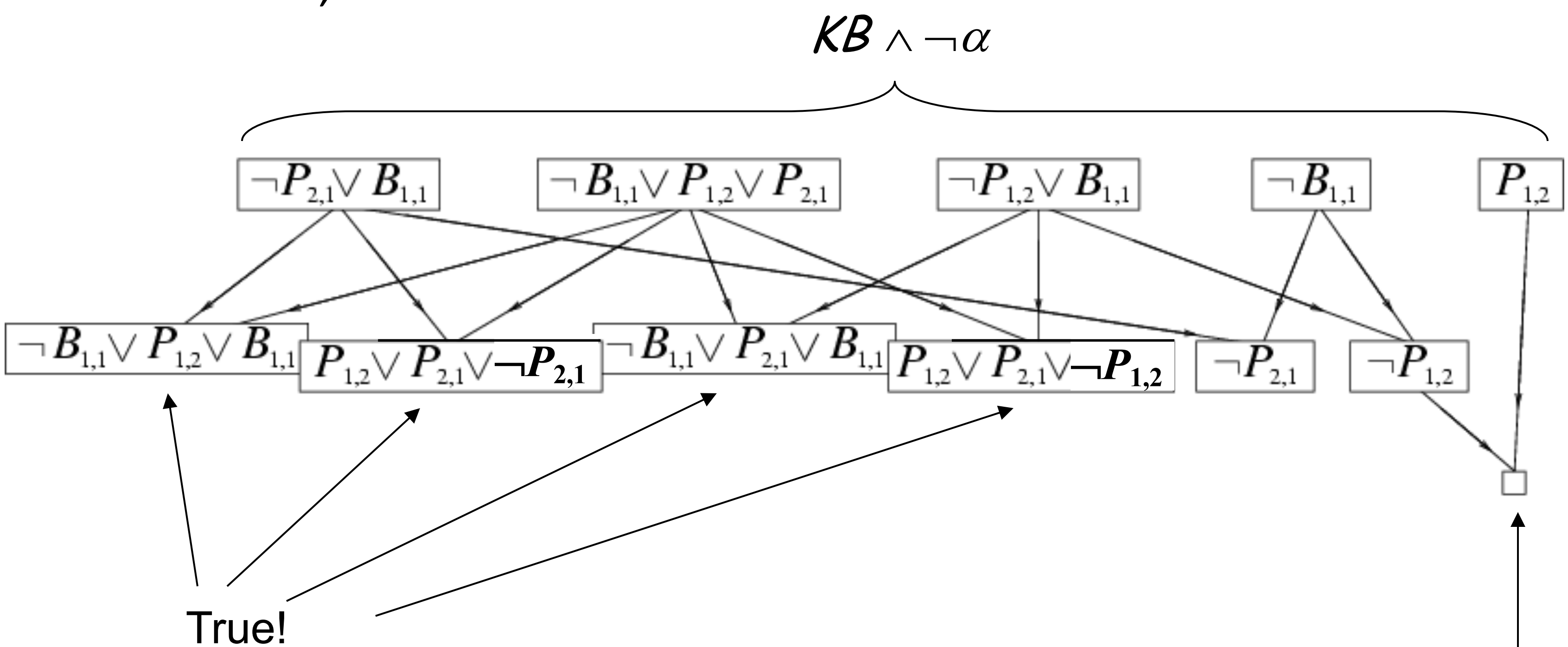
- Resolve $(\neg P_{1,2} \quad B_{1,1})$ and $(\neg\ B_{1,1})$ to give $(\neg P_{1,2})$
- Resolve $(\neg P_{1,2})$ and $(P_{1,2})$ to give $(\ )$

- Consequently, the goal or query sentence is entailed by KB.
- Of course, there are many other proofs, which are OK iff correct.

Resolution example

Graphical view of the proof

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



A sentence in KB is not "used up" when it is used in a resolution step. It is true, remains true, and is still in KB.

False in all worlds

Search

- States: current CNF KB + new clauses
- Operators: resolution
- Initial state: KB + negated goal
- Goal State: a database containing the empty clause
- Search using any search method

Exercise on applying resolution inference procedure

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

Prove:

1. $\Phi_1 \models \Phi_2$, by using the resolution inference procedure

$\Phi_1 \models \Phi_2$, by using the resolution inference procedure.

Write together

Exercise on applying resolution inference procedure

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

Prove:

1. $\Phi_1 \models \Phi_2$, by using the resolution inference procedure

$\Phi_1 \models \Phi_2$, by using the resolution inference procedure.
First of all we have to write Φ_1 and $\neg\Phi_2$ in CNF.

$$\mathrm{CNF}(\Phi_1) = A \wedge (B \vee Q)$$

$$\mathrm{CNF}(\neg\Phi_2) = \neg((A\wedge B)\vee(A\wedge Q)) = \neg(A\wedge B)\wedge\neg(A\wedge Q) = (\neg A\vee\neg B)\wedge(\neg A\vee\neg Q)$$

Then we apply resolution inference:

1. A
2. $B \vee Q$
3. $\neg A \vee \neg B$
4. $\neg A \vee \neg Q$

---

5. $\neg B$ (1,3)
6. $\neg Q$ (1,4)
7. $\neg A \vee Q$ (2,3)
8. $\neg A \vee B$ (2,4)

---

9. Q (1,7)
10. B (1,8)
11. $\neg A$ (3,8)

---

12. $\{\}$ (1,11)

Horn Clauses

- Resolution can be exponential in space and time.

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

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

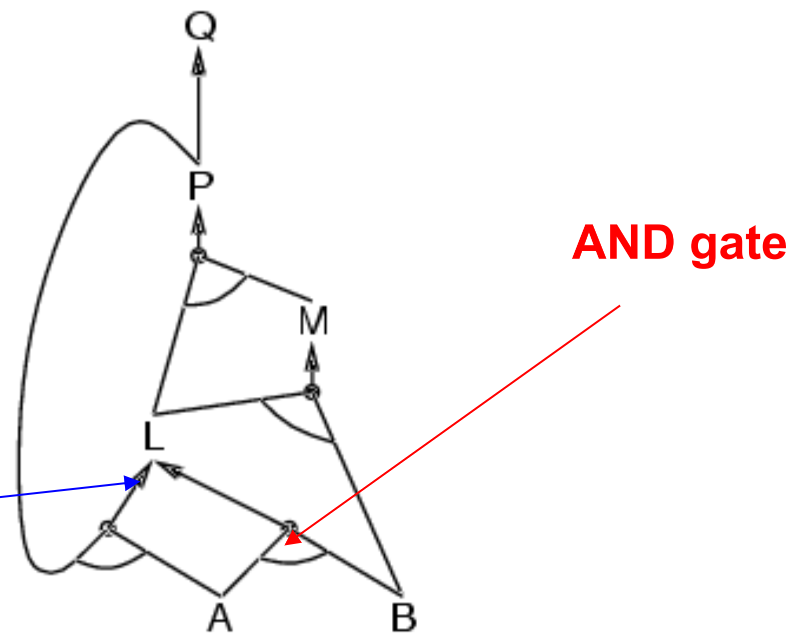
- Every Horn clause can be rewritten as an implication with a conjunction of positive literals in the premises and at most a single positive literal as a conclusion.
  e.g. $A \vee \neg B \vee \neg C \equiv B \wedge C \Rightarrow A$
- 1 positive literal and $\geq$ 1 negative literal: definite clause (e.g., above)
- 0 positive literals: integrity constraint or goal clause
  e.g. $(\neg A \vee \neg B) \equiv (A \wedge B \Rightarrow False)$ states that (A $\wedge$ B) must be false
- 0 negative literals: fact
  e.g., (A) $\equiv$ (True $\Rightarrow$ A) states that A must be true.
- 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.

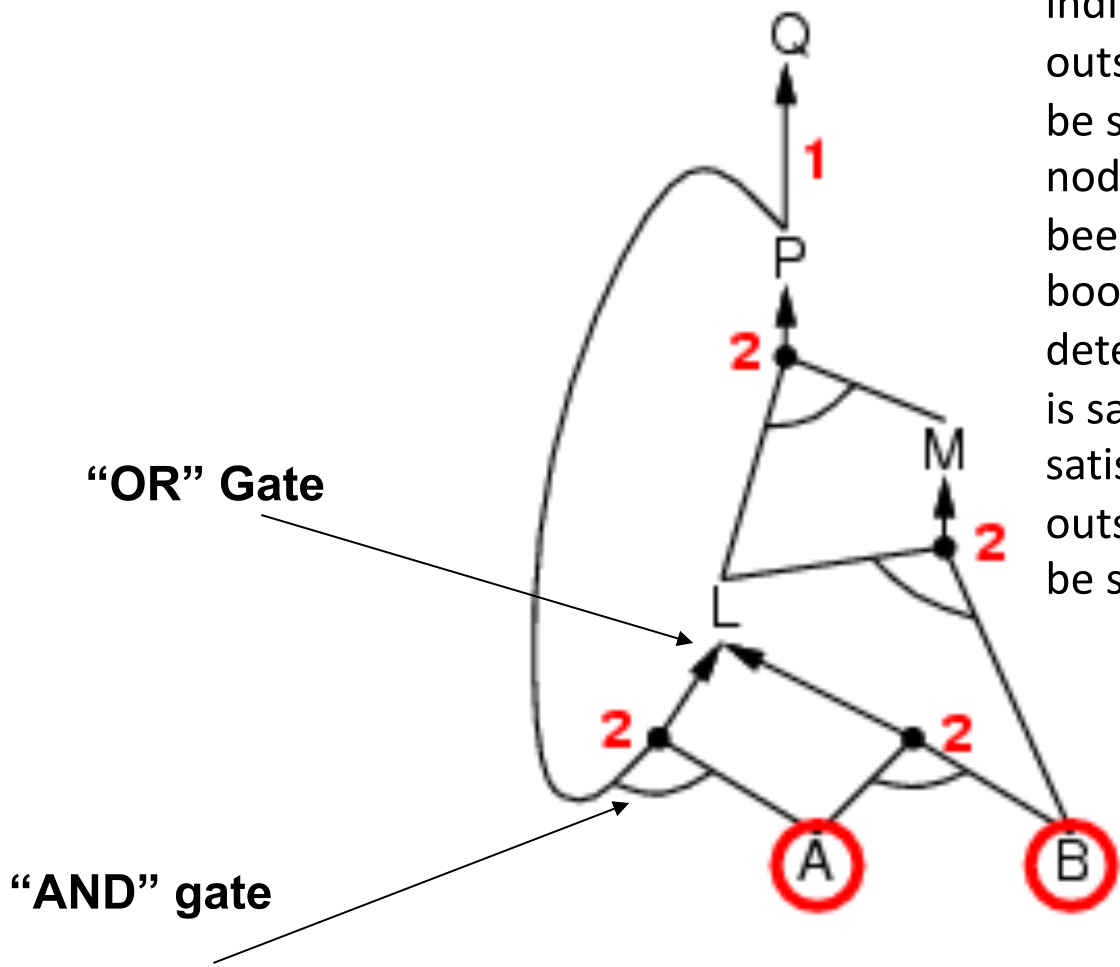
- This proves that *KB* $\Rightarrow$ *Query* is true in all possible worlds, and hence it proves entailment.

$$
\begin{aligned}
&P \Rightarrow Q\\
&L \wedge M \Rightarrow P\\
&B \wedge L \Rightarrow M\\
&A \wedge P \Rightarrow L\\
&A \wedge B \Rightarrow L\\
&A\\
&B
\end{aligned}
$$



- Forward chaining is sound and complete for Horn KB

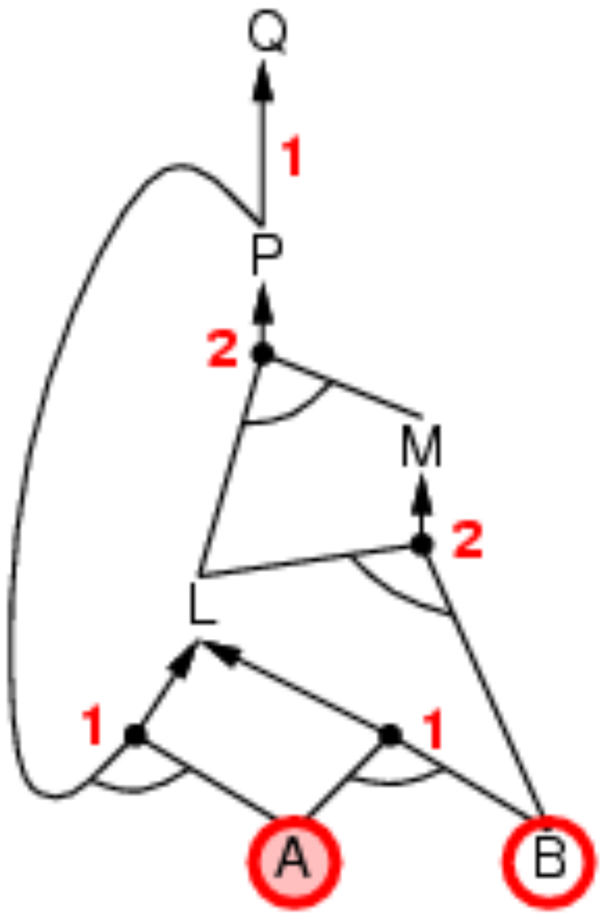
Forward chaining example



Numbers at each AND node indicate the number of outstanding preconditions yet to be satisfied before **all** of that AND node input preconditions have been satisfied. It is an efficient book-keeping mechanism for determining when an AND node is satisfied. The AND node is satisfied when its number of outstanding preconditions yet to be satisfied is zero.

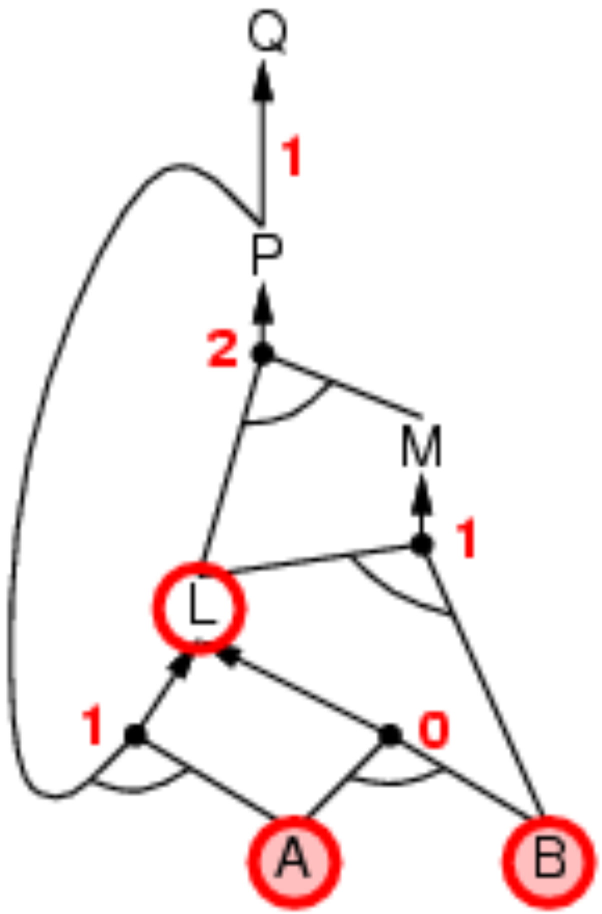
Forward chaining example

$$
\begin{aligned}
&P \Rightarrow Q \\
&L \wedge M \Rightarrow P \\
&B \wedge L \Rightarrow M \\
&A \wedge P \Rightarrow L \\
&A \wedge B \Rightarrow L \\
&A \\
&B
\end{aligned}
$$



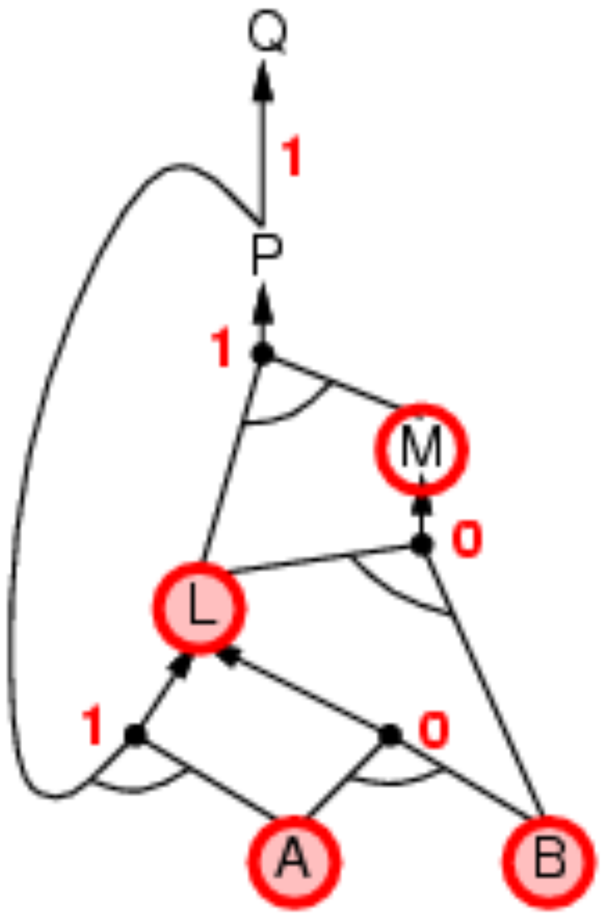
Forward chaining example

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



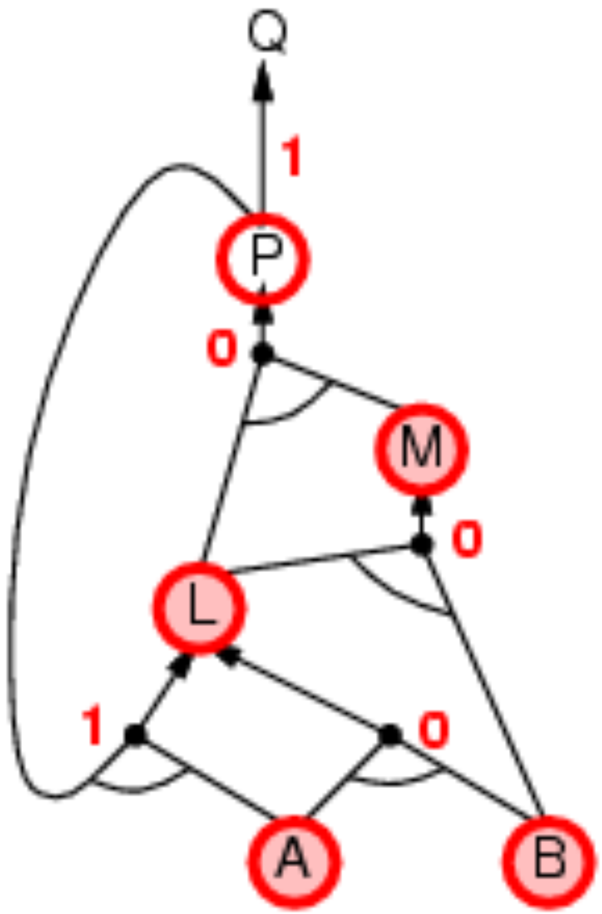
Forward chaining example

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



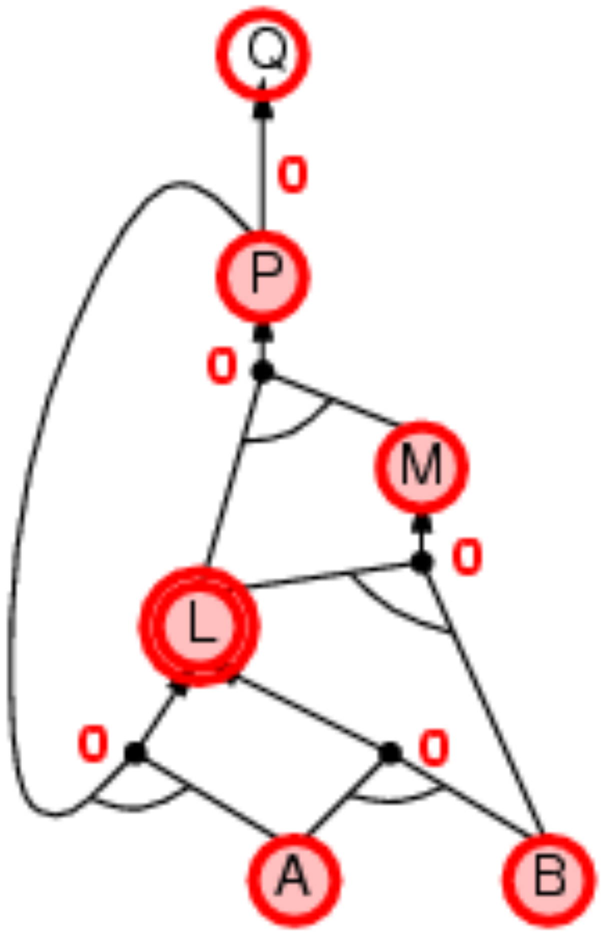
Forward chaining example

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



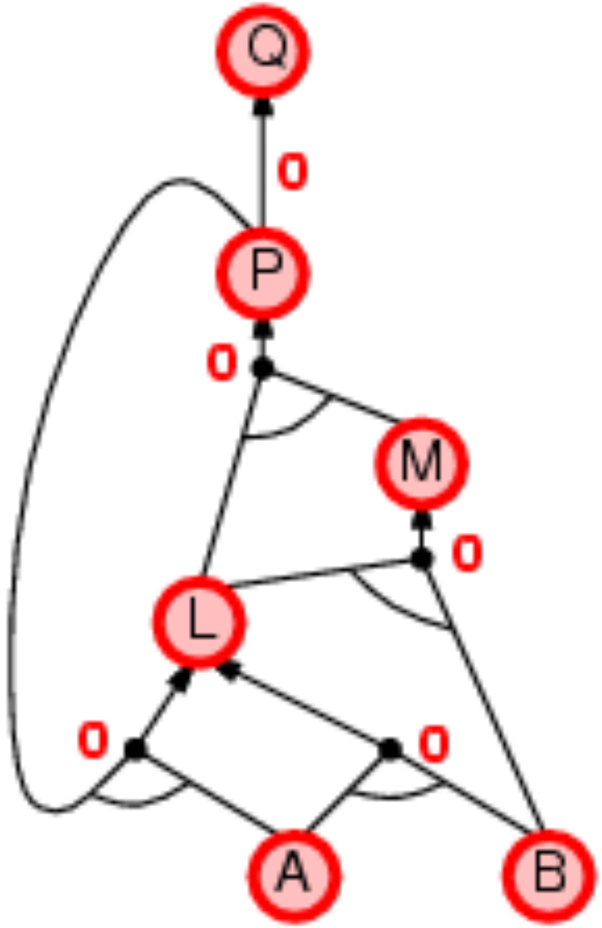
Forward chaining example

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



Forward chaining example

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



FC pseudocode

```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
   local variables: 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 in KB

   while agenda is not empty do
       p ← POP(agenda)
       unless inferred[p] do
           inferred[p] ← 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
```

Backward chaining (BC)

Idea: work backwards from the query *q*

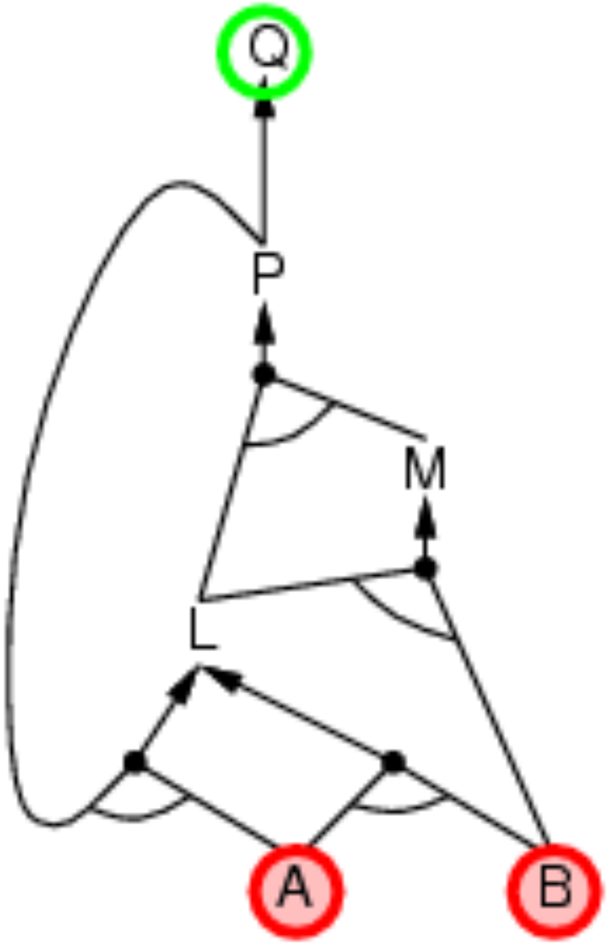
- check if *q* is known already, or
- prove by BC all premises of some rule concluding *q*
- Hence BC maintains a stack of sub-goals that need to be proved to get to q.

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

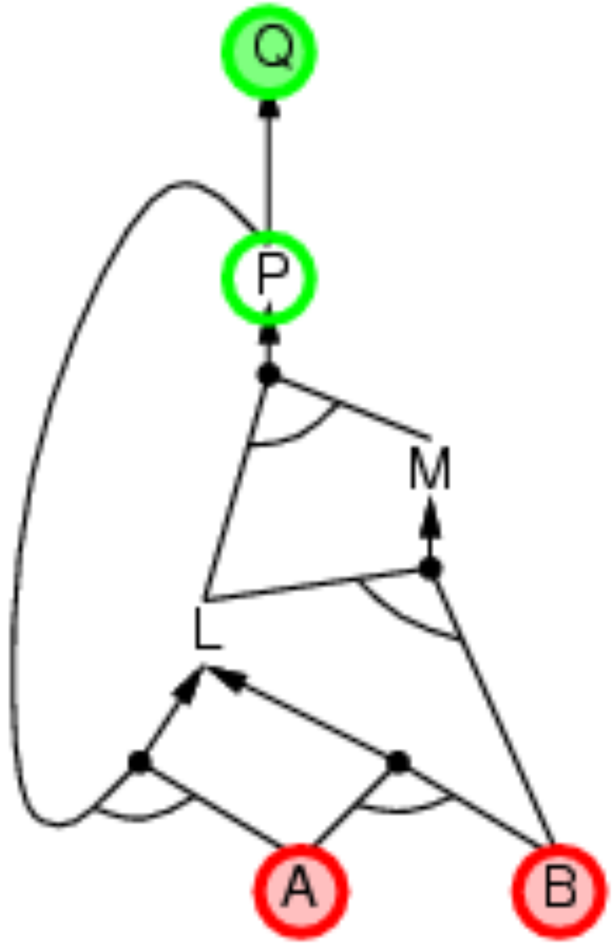
Avoid repeated work: check if new sub-goal

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

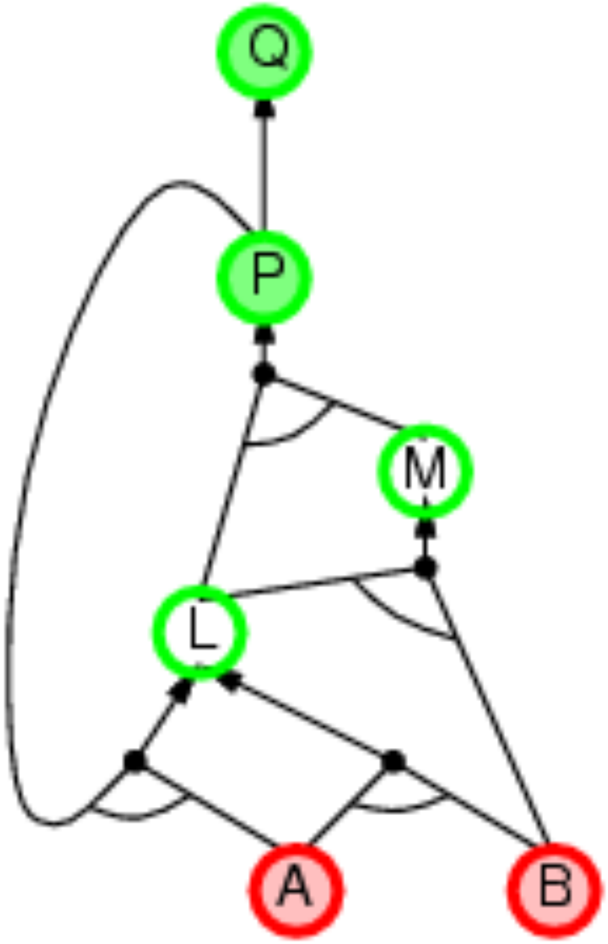
Backward chaining example



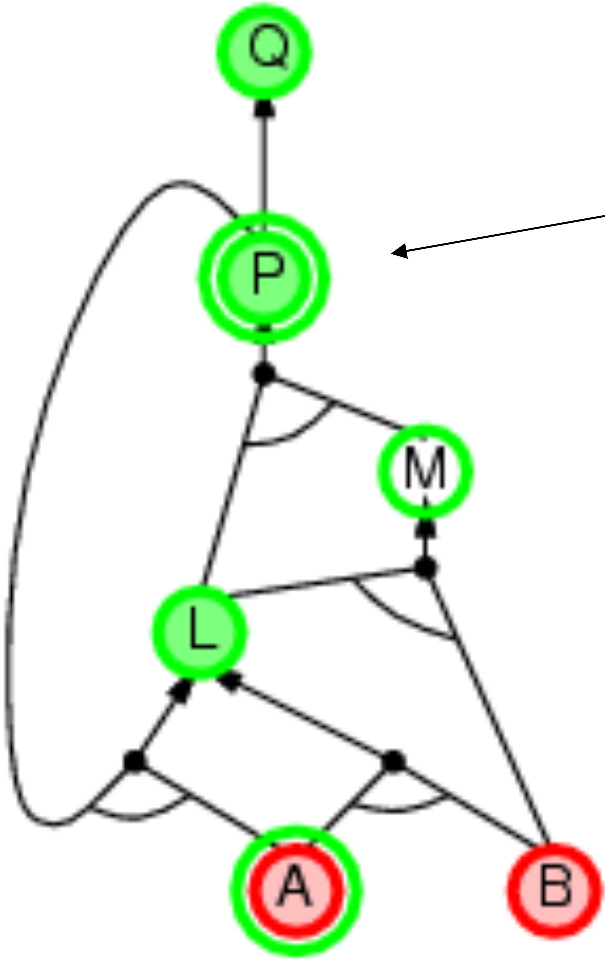
Backward chaining example



Backward chaining example

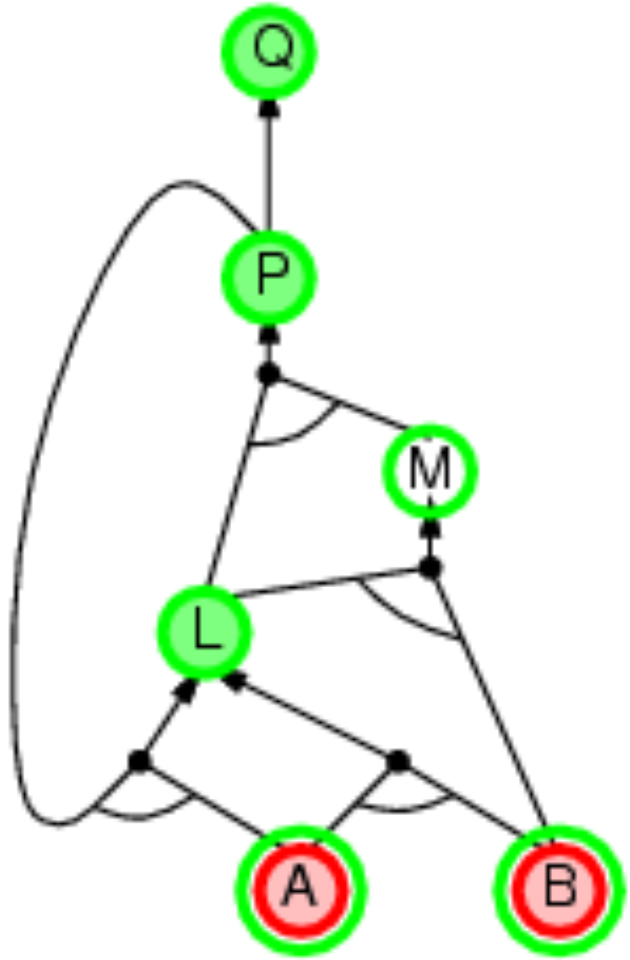


Backward chaining example



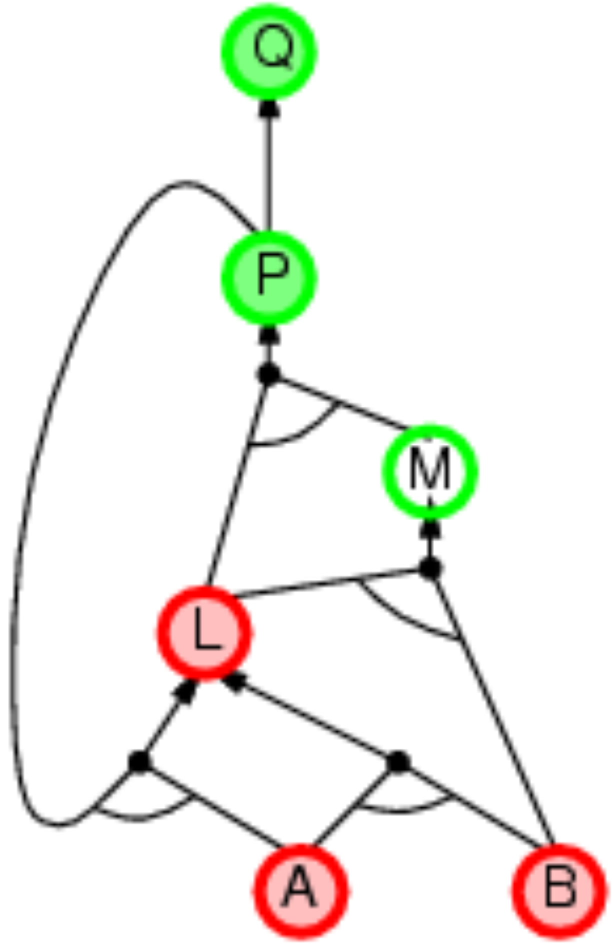
we need P to prove
L and L to prove P.

Backward chaining example

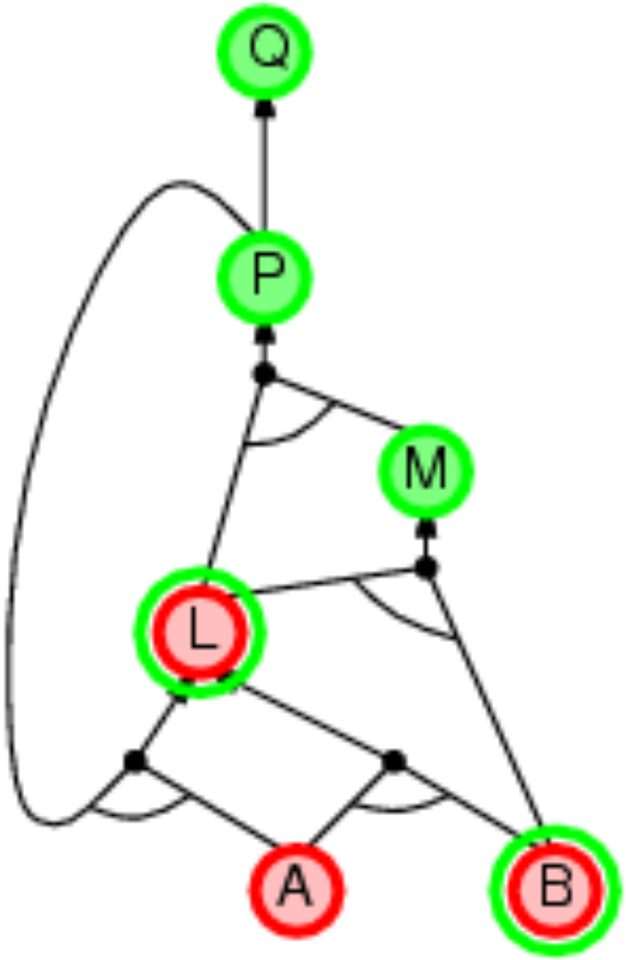


As soon as you can move forward, do so.

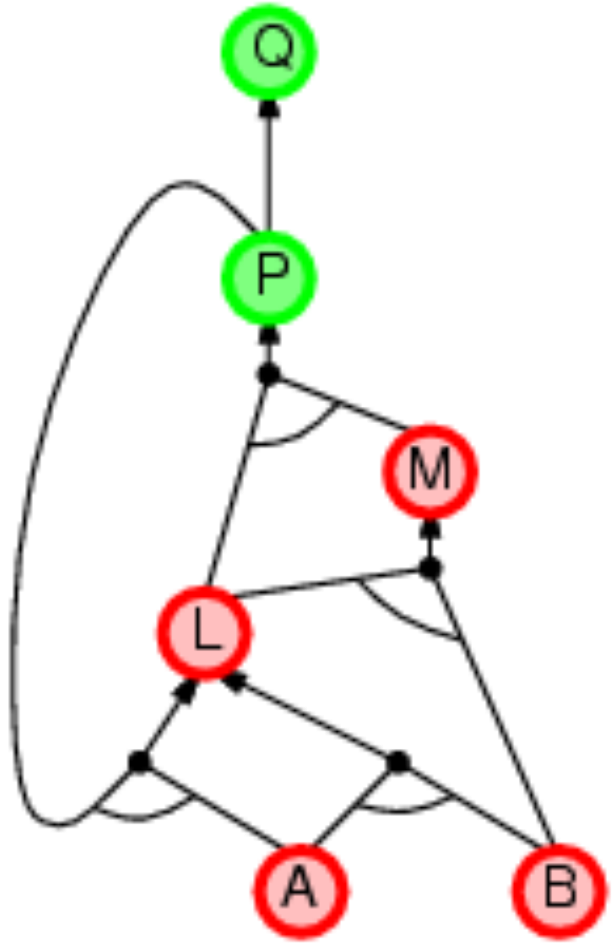
Backward chaining example



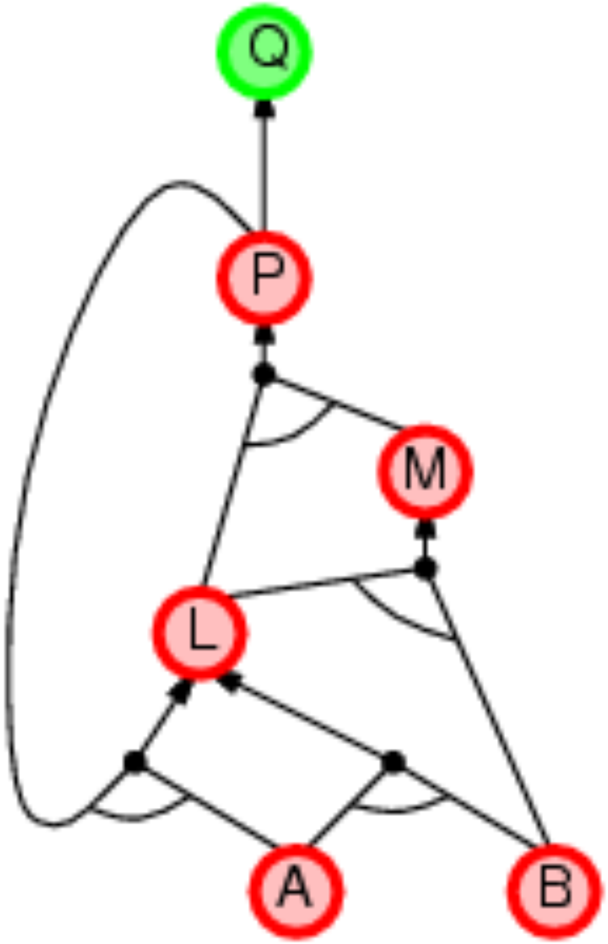
Backward chaining example



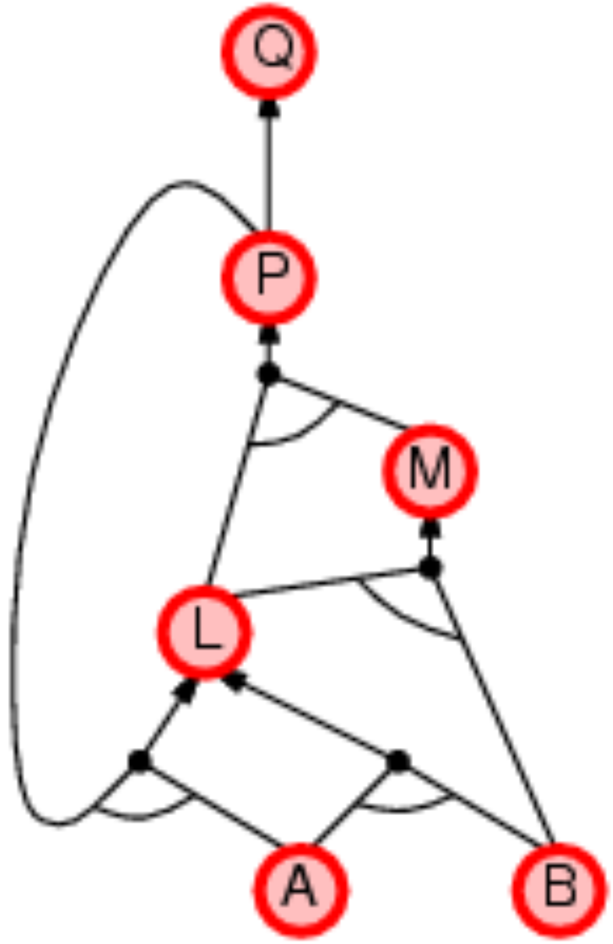
Backward chaining example



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

Algorithms to check satisfiability

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. E.g., $(A \vee B) \wedge (A \vee C)$ is true if A is true

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.

3. Unit clause heuristic
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.
   For example, if the model contains B =true,
   then $(\neg B \vee \neg C)$ simplifies to $\neg C$, which is a unit clause

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

Same exercise solved with the resolution inference procedure with DPLL

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

$\Phi_1 \models \Phi_2$, by using the DPLL algorithm.

First of all we have to write Φ_1 and $\neg\Phi_2$ in CNF.

$$\mathrm{CNF}(\Phi_1) = A \wedge (B \vee Q)$$

$$\mathrm{CNF}(\neg\Phi_2) = \neg((A \wedge B) \vee (A \wedge Q)) = \neg(A \wedge B) \wedge \neg(A \wedge Q) = (\neg A \vee \neg B) \wedge (\neg A \vee \neg Q)$$

1. Early termination
   A clause is true if any literal is true.
   A sentence is false if any clause is false. E.g., (A ∨ B) ∧ (A ∨ C) is true if A is true
2. Pure symbol heuristic
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses (A ∨ ¬B), (¬B ∨ ¬C), (C ∨ A), A and B are pure, C is impure.
3. Unit clause heuristic
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.
   For example, if the model contains B =true,
   then (¬B ∨ ¬C) simplifies to ¬C, which is a unit clause

Same exercise solved with the resolution inference procedure with DPLL

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

$\Phi_1 \models \Phi_2$, by using the DPLL algorithm.

$\{\}$	$A \wedge (A \vee Q) \wedge (A \vee B) \wedge (B \vee Q) \wedge (\neg A \vee \neg B) \wedge (\neg A \vee \neg Q)$	One-Literal on A
$\{A\}$	$(B \vee Q) \wedge (\neg B) \wedge (\neg Q)$	One-Literal on $\neg B$
$\{A, \neg B\}$	$Q \wedge \neg Q$	One-Literal on Q
$\{A, \neg B, Q\}$	$\{\}$	Unsatisfiable

DPLL algorithm

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($clauses, symbols, \{\ \}$)

---

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

  if every clause in $clauses$ is true in $model$ **then return** *true*
  if some clause in $clauses$ is false in $model$ **then return** *false*
  $P, value \leftarrow$ FIND-PURE-SYMBOL($symbols, clauses, model$)
  if P is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P, value \leftarrow$ FIND-UNIT-CLAUSE($clauses, model$)
  if P is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)
  return DPLL($clauses, rest, model \cup \{P{=}true\}$) **or**
      DPLL($clauses, rest, model \cup \{P{=}false\}$))

The WalkSAT algorithm

- Incomplete, local search algorithm
  - In many problems, like the CSP ones, just a consistent assignment is enough
- 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.

Summary

- Resolution inference procedure starting from CNF sentences and the negated query sentence to prove unsatisfiability
- Model checking
  - DPLL: deterministic algorithm based on backtracking search
  - WalkSAT: local search algorithm, to find one satisfiable assignment

Next

- How to make the logic capable of expressing more than facts as in the propositional logic? First order logic