Lecture 7: Logical Agents

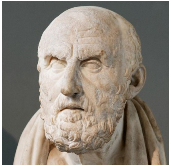
Davide Grossi

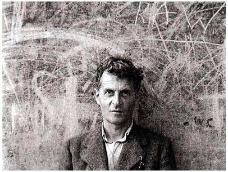
Ariane 5

university of groningen

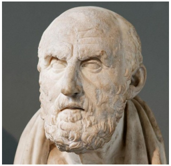
PART I
Propositional Logic Recap

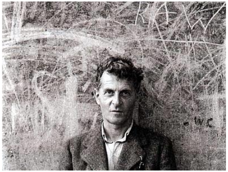
Propositional logic





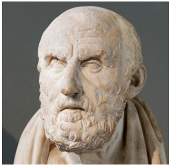
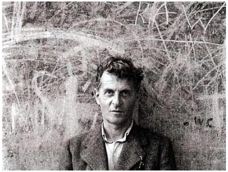
Propositional logic



Propositional logic was born in Ancient Greece with the Stoics (Chrisyppus)

Propositional logic

Propositional logic was born in Ancient Greece with the Stoics (Chrisyppus)
Modern (*mathematical*) logic started with George Boole (*The Mathematical Analysis of Logic*, 1848).

Propositional (or Boolean) logic is what enabled the birth of computer science (Claude Shannon, *A Symbolic Analysis of Relay and Switching Circuits*, 1938).

Syntax of PL: Examples

$\boxed{\text{QUESTION}}$ Which of these are sentences of the language of PL?

- P
- (P)
- $P \vee Q$
- $(((P \Rightarrow Q) \wedge \neg Q) \Rightarrow \neg P)$
- $(())$
- $(P \wedge Q) \vee$
- $(P \neg \Rightarrow Q)$

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?

By means of **truth tables**. Each row corresponds to an **interpretation** of the propositional symbols in a given sentence.

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?
By means of **truth tables**. Each row corresponds to an **interpretation** of the propositional symbols in a given sentence. Each row is said to be a **model** of the sentence.

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?

By means of **truth tables**. Each row corresponds to an **interpretation** of the propositional symbols in a given sentence. Each row is said to be a **model** of the sentence.

$\boxed{\text{QUESTION}}$ What is an interpretation?

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?
By means of **truth tables**. Each row corresponds to an **interpretation** of the propositional symbols in a given sentence. Each row is said to be a **model** of the sentence.

$\boxed{\text{QUESTION}}$ What is an interpretation? An interpretation is an assignment of either 1 or 0 to each propositional symbol in the sentence, and therefore to the sentence itself, following the rules of the logical connectives.

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?
By means of **truth tables**. Each row corresponds to an **interpretation** of the propositional symbols in a given sentence. Each row is said to be a **model** of the sentence.

$\boxed{\text{QUESTION}}$ What is an interpretation? An interpretation is an assignment of either 1 or 0 to each propositional symbol in the sentence, and therefore to the sentence itself, following the rules of the logical connectives.

$\boxed{\text{QUESTION}}$ What are these rules?

Semantics of PL

$\boxed{\text{QUESTION}}$ How is the semantics of PL specified?
By means of **truth tables**. Each row corresponds to an **interpretation** of the propositional symbols in a given sentence. Each row is said to be a **model** of the sentence.

$\boxed{\text{QUESTION}}$ What is an interpretation? An interpretation is an assignment of either 1 or 0 to each propositional symbol in the sentence, and therefore to the sentence itself, following the rules of the logical connectives.

$\boxed{\text{QUESTION}}$ What are these rules?

p	$\neg p$
1	0
0	1

p	q	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

where 1 stands for *true* and 0 for *false*.

Validity, Satisfiability & Logical Consequence

A sentence is **valid** iff $\boxed{?}$

Validity, Satisfiability & Logical Consequence

A sentence is **valid** iff $\boxed{?}$ it is true in *all* models

- a.k.a. **tautologies**
- e.g. $P \vee \neg P$, $P \Rightarrow P$, $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$

Validity, Satisfiability & Logical Consequence

A sentence is **valid** iff $\boxed{?}$ it is true in *all* models

- a.k.a. **tautologies**
- e.g. $P \vee \neg P$, $P \Rightarrow P$, $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$

A sentence is **satisfiable** iff $\boxed{?}$

Validity, Satisfiability & Logical Consequence

A sentence is **valid** iff $\boxed{?}$ it is true in *all* models

- a.k.a. **tautologies**
- e.g. $P \vee \neg P$, $P \Rightarrow P$, $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$

A sentence is **satisfiable** iff $\boxed{?}$ it is true in *at least one* model

- e.g. $P \wedge Q$
- α is not satisfiable iff $\neg\alpha$ is valid (= α is unsatisfiable)

Validity, Satisfiability & Logical Consequence

A sentence is **valid** iff $\boxed{?}$ it is true in *all* models

- a.k.a. **tautologies**
- e.g. $P \lor \neg P$, $P \Rightarrow P$, $(P \land (P \Rightarrow Q)) \Rightarrow Q$

A sentence is **satisfiable** iff $\boxed{?}$ it is true in *at least one* model

- e.g. $P \land Q$
- α is not satisfiable iff $\neg\alpha$ is valid (= α is unsatisfiable)

A sentence α follows logically (is a **logical consequence**) of the set of sentences *KB* iff $\boxed{?}$

Validity, Satisfiability & Logical Consequence

A sentence is **valid** iff $\boxed{?}$ it is true in *all* models

- a.k.a. **tautologies**
- e.g. $P \vee \neg P$, $P \Rightarrow P$, $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$

A sentence is **satisfiable** iff $\boxed{?}$ it is true in *at least one* model

- e.g. $P \wedge Q$
- α is not satisfiable iff $\neg\alpha$ is valid (= α is unsatisfiable)

A sentence α follows logically (is a **logical consequence**) of the set of sentences *KB* iff $\boxed{?}$ α is true in every model in which *KB* is true

- In symbols, $KB \models \alpha$
- In other words, $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$, where $M(\alpha)$ denotes the set of all models that make α true, and $M(KB)$ the set of models that make *KB* true
- …or, *KB* **entails** α iff any model of *KB* is also a model of α

Techniques for checking entailment (i.e., whether $KB \models \alpha$)

- **Model checking**
  - Truth table enumeration. Build all truth-tables for the symbols in *KB* and α and check the entailment

Techniques for checking entailment (i.e., whether $KB \models \alpha$)

- **Model checking**
  - Truth table enumeration. Build all truth-tables for the symbols in *KB* and α and check the entailment
  - Search algorithm for **satisfiability** ($\sim$ search for CSP)

Techniques for checking entailment (i.e., whether $KB \models \alpha$)

- **Model checking**
  - Truth table enumeration. Build all truth-tables for the symbols in KB and α and check the entailment
  - Search algorithm for **satisfiability** ($\sim$ search for CSP)
    - $\boxed{\text{QUESTION}}$ How can you check entailment by checking satisfiability?

Techniques for checking entailment (i.e., whether $KB \models \alpha$)

- **Model checking**
  - Truth table enumeration. Build all truth-tables for the symbols in KB and α and check the entailment
  - Search algorithm for **satisfiability** ($\sim$ search for CSP)
    - $\boxed{\text{QUESTION}}$ How can you check entailment by checking satisfiability?
    - $KB \models \alpha$ iff $(\bigwedge KB) \wedge \neg\alpha$ is unsatisfiable
  - Heuristic search for satisfiability
- **Inference** (or **theorem-proving**)
  - Proof search, that is, construct a chain of conclusions that leads to α (proof) from the premises
  - Can be done by standard search algorithms, where actions are applications of inference rules

PART II
Reasoning in PL: the model-checking way

Brute force: truth table enumeration

function TT-ENTAILS?(KB,α) **return** *true* or *false*
  inputs: KB, a sentence in propositional logic
    α, the query, a sentence in propositional logic
  symbols $\leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL(KB,α, *symbols*,{})

Brute force: truth table enumeration

function TT-ENTAILS?(KB,α) **return** *true* or *false*
  inputs: KB, a sentence in propositional logic
    α, the query, a sentence in propositional logic
  symbols $\leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL(KB,α, *symbols*,{})

function TT-CHECK-ALL(KB,α, *symbols, model*)
  if EMPTY?(*symbols*) **then**
    if PL-TRUE?(KB,*model*) **then**
      return PL-TRUE?(α,*model*)
    else return *true // when KB is false, always return true*
  else do
    P $\leftarrow$ FIRST(*symbols*)
    rest $\leftarrow$ REST(*symbols*)
    return (TT-CHECK-ALL(KB,α, *rest* ,*model* $\cup \{P\}$)
     and (TT-CHECK-ALL(KB,α, *rest* ,*model* $\cup \{\neg P\}$))

Brute force: truth table enumeration

function TT-ENTAILS?(KB,α) **return** *true* or *false*
  inputs: KB, a sentence in propositional logic
    α, the query, a sentence in propositional logic
  symbols $\leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL(KB,α, *symbols*,{})

function TT-CHECK-ALL(KB,α, *symbols, model*)
  if EMPTY?(*symbols*) **then**
    if PL-TRUE?(KB,*model*) **then**
      return PL-TRUE?(α,*model*)
    else return *true // when KB is false, always return true*
  else do
    P $\leftarrow$ FIRST(*symbols*)
    rest $\leftarrow$ REST(*symbols*)
    return (TT-CHECK-ALL(KB,α, *rest* ,*model* $\cup\{P\}$)
      and (TT-CHECK-ALL(KB,α, *rest* ,*model* $\cup\{\neg P\}$))

$\boxed{\text{QUESTION}}$ How does the algorithm perform?

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff ? it is a conjunction of clauses that are disjunctions of literals (i.e., atomic propositions or negations thereof)

- e.g. $(P_1 \vee P_2 \vee \neg P_3) \wedge (\neg P_1 \vee \neg P_2)$

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff ⍰ it is a conjunction of clauses that are disjunctions of literals (i.e., atomic propositions or negations thereof)

- e.g. $(P_1 \vee P_2 \vee \neg P_3) \wedge (\neg P_1 \vee \neg P_2)$

Any sentence (e.g. $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$) can be turned into a CNF sentence

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff ? it is a conjunction of clauses that are disjunctions of literals (i.e., atomic propositions or negations thereof)

- e.g. $(P_1 \vee P_2 \vee \neg P_3) \wedge (\neg P_1 \vee \neg P_2)$

Any sentence (e.g. $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$) can be turned into a CNF sentence

1. Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
   - $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff ⍰ it is a conjunction of clauses that are disjunctions of literals (i.e., atomic propositions or negations thereof)

- e.g. $(P_1 \vee P_2 \vee \neg P_3) \wedge (\neg P_1 \vee \neg P_2)$

Any sentence (e.g. $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$) can be turned into a CNF sentence

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 $\alpha \Rightarrow \beta$ by $\neg\alpha \vee \beta$
   - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff it is a conjunction of clauses that are disjunctions of literals (i.e., atomic propositions or negations thereof)

- e.g. $(P_1 \vee P_2 \vee \neg P_3) \wedge (\neg P_1 \vee \neg P_2)$

Any sentence (e.g. $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$) can be turned into a CNF sentence

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 $\alpha \Rightarrow \beta$ by $\neg\alpha \vee \beta$
   - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. Move $\neg$ inside brackets using DeMorgan's laws
   - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

Trick for Checking Satisfiability: Conjunctive Normal Form

A sentence is in **conjunctive normal form** (CNF) iff it is a conjunction of clauses that are disjunctions of literals (i.e., atomic propositions or negations thereof)

- e.g. $(P_1 \lor P_2 \lor \neg P_3) \land (\neg P_1 \lor \neg P_2)$

Any sentence (e.g. $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$) can be turned into a CNF sentence

1. **Eliminate** $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$
   - $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ by $\neg\alpha \lor \beta$
   - $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
3. Move $\neg$ inside brackets using DeMorgan's laws
   - $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$
4. Distribute $\lor$ over $\land$
   - $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

An algorithm to check satisfiability

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

- Determines satisfiability of a CNF sentence
- Recursive, depth-first enumeration of possible models (like in TT-ENTAILS?(KB,α)), with three improvements

An algorithm to check satisfiability

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

- Determines satisfiability of a CNF sentence
- Recursive, depth-first enumeration of possible models (like in TT-ENTAILS?(KB,α)), with three improvements
  1. *Early termination*:
     A clause is true if any literal is true. A sentence is true if all clauses are true and is false if any clause is false.

An algorithm to check satisfiability

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

- Determines satisfiability of a CNF sentence
- Recursive, depth-first enumeration of possible models (like in TT-ENTAILS?(KB,α)), with three improvements
  1. *Early termination*:
     A clause is true if any literal is true. A sentence is true if all clauses are true and is false if any clause is false.
  2. *Pure symbol heuristic*:
     A **pure symbol** appears with the same "sign" in all clauses. e.g., In the three clauses $(A \vee \neg B), (\neg B \vee C), (\neg C \vee A)$, A and B are pure, C is impure. DPLL makes a pure symbol literal true.

An algorithm to check satisfiability

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

- Determines satisfiability of a CNF sentence
- Recursive, depth-first enumeration of possible models (like in TT-ENTAILS?(KB,α)), with three improvements
  1. *Early termination*:
     A clause is true if any literal is true. A sentence is true if all clauses are true and is false if any clause is false.
  2. *Pure symbol heuristic*:
     A **pure symbol** appears with the same "sign" in all clauses. e.g., In the three clauses $(A \vee \neg B), (\neg B \vee C), (\neg C \vee A)$, A and B are pure, C is impure. DPLL makes a pure symbol literal true.
     QUESTION Why would you do that?
  3. *Unit clause heuristic*:
     A **unit clause** only has one literal. In DPLL, clauses in which only one literal is not assigned *false* are also called unit clauses.
     DPLL makes literals in unit clauses true.

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

function DPLL-SATISFIABLE?(*s*) **returns** *true/false*
  clauses $\leftarrow$ the set of CNF clauses of *s*
  symbols $\leftarrow$ a list of the proposition symbols in *s*
  return DPLL(*clauses, symbols*,{ })

Davis-Putnam-Logemann-Loveland (DPLL) algorithm

function DPLL-SATISFIABLE?(*s*) **returns** *true/false*
    clauses ← the set of CNF clauses of *s*
    symbols ← a list of the proposition symbols in *s*
    return DPLL(*clauses, symbols*,{ })

function DPLL(*clauses, symbols, model*) **returns** *true/false*
    if every clause in *clauses* is *true* in *model* then **return** *true*
    if a clause in *clauses* is *false* in *model* then **return** *false*
    P, val ← FIND-PURE-SYMBOL(*symbols, clauses, model*)
    if *P* is non-null **then**
        return DPLL (*clauses, symbols - P, model* ∪ {*P=val*})
    P, val ← FIND-UNIT-CLAUSE(*symbols, clauses, model*)
    if *P* is non-null **then**
        return DPLL (*clauses, symbols - P, model* ∪ {*P=val*})
    P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
    return DPLL(*clauses, rest, model* ∪ {*P=true*}) **or**
        DPLL(*clauses, rest, model* ∪ {*P=false*})

SAT solvers

SAT solvers extend the basics of the DPLL algorithm in various ways in order to scale up to large problems, e.g.:

- Variable and value ordering
- Intelligent backtracking
- Random restarts
- ... and more

SAT solvers

SAT solvers extend the basics of the DPLL algorithm in various ways in order to scale up to large problems, e.g.:

- Variable and value ordering
- Intelligent backtracking
- Random restarts
- …and more

An alternative is based on local search

- Objective function: minimize the number of unsatisfied clauses
- Limitation: cannot detect unsatisfiability in general

SAT solvers

SAT solvers extend the basics of the DPLL algorithm in various ways in order to scale up to large problems, e.g.:

- Variable and value ordering
- Intelligent backtracking
- Random restarts
- ... and more

An alternative is based on local search

- Objective function: minimize the number of unsatisfied clauses
- Limitation: cannot detect unsatisfiability in general
- QUESTION Why?

WalkSAT

function WALKSAT(*clauses, p, max_flips*) **returns** model
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of a random move,
    max_flips, number of flips before giving up

  model $\leftarrow$ a random assignment of symbols in *clauses*
  for $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause $\leftarrow$ random clause from *clauses*, false in *model*
    with probability *p*
      flip value of random symbol in *clause*
    else flip the symbol in *clause* that
      maximizes number of satisfied clauses
  return *failure*

Easy and hard problems

Some satisfiability problems are easier than others

- **Underconstrained** problems are easy
  - They have few clauses relative to the available symbols
  - e.g. $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$
  - In this example with 5 clauses and 5 variables, 16 out of 32 assignments are solutions
- **Overconstrained** problems are likely to have no solution

PART III
Reasoning in PL: the theorem-proving way

Inference rules

A proof for α is a chain of conclusions ending with α

- Conclusions are the result of inference rules like:

Inference rules

A proof for α is a chain of conclusions ending with α

- Conclusions are the result of inference rules like:
  - **Modus ponens**

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

Inference rules

A proof for α is a chain of conclusions ending with α

- Conclusions are the result of inference rules like:
  - **Modus ponens**

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

  - **And elimination**

$$\frac{\alpha \land \beta}{\alpha}$$

Inference rules

A proof for α is a chain of conclusions ending with α

- Conclusions are the result of inference rules like:
  - **Modus ponens**

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

  - **And elimination**

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

  - **Contraposition**

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

Inference rules

A proof for α is a chain of conclusions ending with α

- Conclusions are the result of inference rules like:
  - **Modus ponens**

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

  - **And elimination**

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

  - **Contraposition**

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

- Proof search can be more efficient than enumeration

Inference rules

A proof for α is a chain of conclusions ending with α

- Conclusions are the result of inference rules like:
  - **Modus ponens**

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

  - **And elimination**

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

  - **Contraposition**

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

- Proof search can be more efficient than enumeration They typically ignore irrelevant propositions

Proof by resolution: intuition

- From any two clauses (of a CNF) containing complementary literals (i.e., one is the negation of the other) infer a new clause (the so-called **resolvent**) not containing the complementary literals
- Apply such rule on all clauses of the *KB* until either there are no more complementary literals or you obtained an empty resolvent
- If an empty resolvent is obtained, that means the CNF is not satisfiable. Otherwise a model of the CNF can be built out of the simplified clauses

Proof by resolution

- **Unit resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

where ℓ_i and m are complementary literals.

Proof by resolution

- **Unit resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

where ℓ_i and m are complementary literals.

- (Full) **Resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where ℓ_i and m_j are complementary literals.

Proof by resolution

- **Unit resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

  where ℓ_i and m are complementary literals.

- (Full) **Resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  where ℓ_i and m_j are complementary literals.

- $\boxed{\text{QUESTION}}$ Are these rule sound?

Proof by resolution

- **Unit resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

where ℓ_i and m are complementary literals.

- (Full) **Resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where ℓ_i and m_j are complementary literals.

- QUESTION Are these rule sound?
- Yes! So resolution is a sound proof method. What about completeness?

university of groningen

Proof by resolution

- **Unit resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

  where ℓ_i and m are complementary literals.

- (Full) **Resolution**

$$\frac{\ell_1 \vee \cdots \vee \ell_k \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  where ℓ_i and m_j are complementary literals.

- $\boxed{\text{QUESTION}}$ Are these rule sound?
- Yes! So resolution is a sound proof method. What about completeness? Yes it is! The argument to show that is complex but not difficult (see book)

Soundness and completeness (recap)

- A **sound** inference algorithm only derives entailed sentences
  - $KB \vdash_i \alpha$ implies $KB \models \alpha$
  - Unsound inference algorithms make unsupported conclusions

Soundness and completeness (recap)

- A **sound** inference algorithm only derives entailed sentences
  - $KB \vdash_i \alpha$ implies $KB \models \alpha$
  - Unsound inference algorithms make unsupported conclusions
- A **complete** inference algorithm derives any entailed sentence
  - $KB \models \alpha$ implies $KB \vdash_i \alpha$

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0	1				
1	0	1	1				
1	1	0	1				
1	1	1	1				

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0					
0	0	1					
0	1	0	1				
0	1	1	1				
1	0	0	1				
1	0	1	1				
1	1	0	1				
1	1	1	1				

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0				
0	0	1	0				
0	1	0	1				
0	1	1	1				
1	0	0	1				
1	0	1	1				
1	1	0	1				
1	1	1	1				

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1			
0	0	1	0	1			
0	1	0	1	1			
0	1	1	1	1			
1	0	0	1				
1	0	1	1				
1	1	0	1				
1	1	1	1				

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1			
0	0	1	0	1			
0	1	0	1	1			
0	1	1	1	1			
1	0	0	1				
1	0	1	1	1			
1	1	0	1				
1	1	1	1	1			

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1			
0	0	1	0	1			
0	1	0	1	1			
0	1	1	1	1			
1	0	0	1	0			
1	0	1	1	1			
1	1	0	1	0			
1	1	1	1	1			

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1	0		
0	0	1	0	1	0		
0	1	0	1	1	1		
0	1	1	1	1	1		
1	0	0	1	0	0		
1	0	1	1	1	1		
1	1	0	1	0	0		
1	1	1	1	1	1		

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1	0		
0	0	1	0	1	0		
0	1	0	1	1	1	1	
0	1	1	1	1	1	1	
1	0	0	1	0	0		
1	0	1	1	1	1		
1	1	0	1	0	0	1	
1	1	1	1	1	1	1	

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1	0		
0	0	1	0	1	0	1	
0	1	0	1	1	1	1	
0	1	1	1	1	1	1	
1	0	0	1	0	0		
1	0	1	1	1	1	1	
1	1	0	1	0	0	1	
1	1	1	1	1	1	1	

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1	0	0	
0	0	1	0	1	0	1	
0	1	0	1	1	1	1	
0	1	1	1	1	1	1	
1	0	0	1	0	0	0	
1	0	1	1	1	1	1	
1	1	0	1	0	0	1	
1	1	1	1	1	1	1	

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	1	1	1	1	
0	1	1	1	1	1	1	
1	0	0	1	0	0	0	1
1	0	1	1	1	1	1	
1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	

Soundness of resolution: an example

$$\frac{P \vee Q \qquad \neg P \vee R}{Q \vee R}$$

P	Q	R	α $P \vee Q$	β $\neg P \vee R$	$\alpha \wedge \beta$	γ $Q \vee R$	$(\alpha \wedge \beta) \Rightarrow \gamma$
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	1
1	0	1	1	1	1	1	1
1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	1

Proof by resolution: more details

- The conclusion of a resolution step is a **resolvent**
- No duplicate terms in resolvents (called factoring)

$$\frac{P \vee Q \vee R \vee S \qquad \neg P \vee Q \vee W}{Q \vee R \vee S \vee W}$$

- Resolve *only* on single pairs of complementary literals

$$\frac{P \vee Q \vee \neg R \qquad W \vee \neg Q \vee R}{P \vee \neg R \vee R \vee W}$$

Proof by resolution: more details

- The conclusion of a resolution step is a **resolvent**
- No duplicate terms in resolvents (called factoring)

$$\frac{P \vee Q \vee R \vee S \qquad \neg P \vee Q \vee W}{Q \vee R \vee S \vee W}$$

- Resolve *only* on single pairs of complementary literals

$$\frac{P \vee Q \vee \neg R \qquad W \vee \neg Q \vee R}{P \vee \neg R \vee R \vee W}$$

and not on multiple ones as in

$$\frac{P \vee Q \vee \neg R \qquad W \vee \neg Q \vee R}{P \vee W}$$

Proof by resolution: more details

- The conclusion of a resolution step is a **resolvent**
- No duplicate terms in resolvents (called factoring)

$$\frac{P \lor Q \lor R \lor S \qquad \neg P \lor Q \lor W}{Q \lor R \lor S \lor W}$$

- Resolve *only* on single pairs of complementary literals

$$\frac{P \lor Q \lor \neg R \qquad W \lor \neg Q \lor R}{P \lor \neg R \lor R \lor W}$$

and not on multiple ones as in

$$\frac{P \lor Q \lor \neg R \qquad W \lor \neg Q \lor R}{P \lor W}$$

- $\boxed{\text{QUESTION}}$ Why?

Proof by resolution: more details

- The conclusion of a resolution step is a **resolvent**
- No duplicate terms in resolvents (called factoring)

$$\frac{P \vee Q \vee R \vee S \qquad \neg P \vee Q \vee W}{Q \vee R \vee S \vee W}$$

- Resolve *only* on single pairs of complementary literals

$$\frac{P \vee Q \vee \neg R \qquad W \vee \neg Q \vee R}{P \vee \neg R \vee R \vee W}$$

  and not on multiple ones as in

$$\frac{P \vee Q \vee \neg R \qquad W \vee \neg Q \vee R}{P \vee W}$$

- $\boxed{\text{QUESTION}}$ Why?
- It would generate unsound inferences such as:

$$\frac{P \vee \neg Q \qquad \neg P \vee Q}{\emptyset}$$

  while $(P \vee \neg Q) \wedge (\neg P \vee Q)$ is clearly satisfiable!

Resolution algorithm for checking entailment (i.e., $KB \models \alpha$?)

1. Convert the knowledge base *KB* into CNF
2. Resolve $KB \wedge \neg\alpha$. Loop until there are no more resolvable pairs:
   2.1 Resolve a pair containing complementary literals
   2.2 If the resolvent is the empty clause, $KB \wedge \neg\alpha$ is unsatisfiable. Conclude that $KB \models \alpha$.
   2.3 Otherwise add the resolvent to *KB*
3. Conclude that $KB \not\models \alpha$

Resolution algorithm for checking entailment (i.e., $KB \models \alpha$?)

1. Convert the knowledge base KB into CNF
2. Resolve $KB \wedge \neg\alpha$. Loop until there are no more resolvable pairs:
   - **2.1** Resolve a pair containing complementary literals
   - **2.2** If the resolvent is the empty clause, $KB \wedge \neg\alpha$ is unsatisfiable. Conclude that $KB \models \alpha$.
   - **2.3** Otherwise add the resolvent to KB
3. Conclude that $KB \not\models \alpha$

$\boxed{\text{QUESTION}}$ We check whether $KB \models \alpha$ by checking whether $KB \wedge \neg\alpha$ is satisfiable. Why does this work?

Resolution

```
function PL-RESOLUTION(KB,α) returns true/false
    inputs: KB, a sentence in propositional logic
            α, the query sentence in propositional logic

    clauses ← the set of CNF clauses 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 empty cl. then return true
            new ← new ∪ resolvents
        if new ⊆ clauses then return false
        clauses ← clauses ∪ new
```

Examples

$$\textit{rain} \lor \textit{sprinklers}, \neg \textit{rain} \models \textit{sprinklers}$$

- KB in CNF is $(\textit{rain} \lor \textit{sprinklers}) \land \neg \textit{rain}$
- Show that $(\textit{rain} \lor \textit{sprinklers}) \land \neg \textit{rain} \land \neg \textit{sprinklers}$ is not satisfiable
- There are therefore three clauses on which to apply resolution: $(\textit{rain} \lor \textit{sprinklers}), \neg \textit{rain}, \neg \textit{sprinklers}$

Examples

$$\textit{rain} \vee \textit{sprinklers}, \neg \textit{rain} \models \textit{sprinklers}$$

- $\textit{KB}$ in CNF is $(\textit{rain} \vee \textit{sprinklers}) \wedge \neg \textit{rain}$
- Show that $(\textit{rain} \vee \textit{sprinklers}) \wedge \neg \textit{rain} \wedge \neg \textit{sprinklers}$ is not satisfiable
- There are therefore three clauses on which to apply resolution: $(\textit{rain} \vee \textit{sprinklers}), \neg \textit{rain}, \neg \textit{sprinklers}$

$$\frac{\textit{rain} \vee \textit{sprinklers} \qquad \neg \textit{rain}}{\textit{sprinklers}}$$

Examples

$$\textit{rain} \lor \textit{sprinklers}, \neg \textit{rain} \models \textit{sprinklers}$$

- $\textit{KB}$ in CNF is $(\textit{rain} \lor \textit{sprinklers}) \land \neg \textit{rain}$
- Show that $(\textit{rain} \lor \textit{sprinklers}) \land \neg \textit{rain} \land \neg \textit{sprinklers}$ is not satisfiable
- There are therefore three clauses on which to apply resolution: $(\textit{rain} \lor \textit{sprinklers}), \neg \textit{rain}, \neg \textit{sprinklers}$

$$\cfrac{\cfrac{\textit{rain} \lor \textit{sprinklers} \qquad \neg \textit{rain}}{\textit{sprinklers}} \qquad \neg \textit{sprinklers}}{\emptyset}$$

Examples

$$\textit{rain} \vee \textit{sprinklers}, \neg \textit{rain} \models \textit{sprinklers}$$

- KB in CNF is $(\textit{rain} \vee \textit{sprinklers}) \wedge \neg \textit{rain}$
- Show that $(\textit{rain} \vee \textit{sprinklers}) \wedge \neg \textit{rain} \wedge \neg \textit{sprinklers}$ is not satisfiable
- There are therefore three clauses on which to apply resolution: $(\textit{rain} \vee \textit{sprinklers}), \neg \textit{rain}, \neg \textit{sprinklers}$

$$\dfrac{\dfrac{\textit{rain} \vee \textit{sprinklers} \qquad \neg \textit{rain}}{\textit{sprinklers}} \qquad \neg \textit{sprinklers}}{\emptyset}$$

or

$$\dfrac{\textit{rain} \vee \textit{sprinklers} \qquad \neg \textit{sprinklers}}{\textit{rain}}$$

Examples

$$rain \lor sprinklers, \neg rain \models sprinklers$$

- KB in CNF is $(rain \lor sprinklers) \land \neg rain$
- Show that $(rain \lor sprinklers) \land \neg rain \land \neg sprinklers$ is not satisfiable
- There are therefore three clauses on which to apply resolution: $(rain \lor sprinklers), \neg rain, \neg sprinklers$

$$\dfrac{\dfrac{rain \lor sprinklers \qquad \neg rain}{sprinklers} \qquad \neg sprinklers}{\emptyset}$$

or

$$\dfrac{\dfrac{rain \lor sprinklers \qquad \neg sprinklers}{rain} \qquad \neg rain}{\emptyset}$$

Jump to Horn

Examples

Modus ponens: $P, P \Rightarrow Q \models Q$

- Show that the set $\{P, P \Rightarrow Q, \neg Q\}$ is not satisfiable
- In CNF: $P \wedge (\neg P \vee Q) \wedge \neg Q$

Examples

Modus ponens: $P, P \Rightarrow Q \models Q$

- Show that the set $\{P, P \Rightarrow Q, \neg Q\}$ is not satisfiable
- In CNF: $P \wedge (\neg P \vee Q) \wedge \neg Q$

$$\frac{\neg P \vee Q \qquad P}{Q}$$

Examples

Modus ponens: $P, P \Rightarrow Q \models Q$

- Show that the set $\{P, P \Rightarrow Q, \neg Q\}$ is not satisfiable
- In CNF: $P \wedge (\neg P \vee Q) \wedge \neg Q$

$$\dfrac{\dfrac{\neg P \vee Q \qquad P}{Q} \qquad \neg Q}{\emptyset}$$

Examples

Is $\neg A \wedge (B \vee C) \wedge (\neg C \vee \neg D) \wedge (\neg B \vee \neg D) \wedge (A \vee D)$ satisfiable?

Examples

Is $\neg A \wedge (B \vee C) \wedge (\neg C \vee \neg D) \wedge (\neg B \vee \neg D) \wedge (A \vee D)$ satisfiable?

$$\frac{B \vee C \qquad \neg C \vee \neg D}{B \vee \neg D}$$

Examples

Is $\neg A \wedge (B \vee C) \wedge (\neg C \vee \neg D) \wedge (\neg B \vee \neg D) \wedge (A \vee D)$ satisfiable?

$$\dfrac{\dfrac{B \vee C \qquad \neg C \vee \neg D}{B \vee \neg D} \qquad \neg B \vee \neg D}{\neg D}$$

Examples

Is $\neg A \wedge (B \vee C) \wedge (\neg C \vee \neg D) \wedge (\neg B \vee \neg D) \wedge (A \vee D)$ satisfiable?

$$\cfrac{\cfrac{\cfrac{B \vee C \qquad \neg C \vee \neg D}{B \vee \neg D} \qquad \neg B \vee \neg D}{\neg D} \qquad A \vee D}{A}$$

Examples

Is $\neg A \land (B \lor C) \land (\neg C \lor \neg D) \land (\neg B \lor \neg D) \land (A \lor D)$ satisfiable?

$$\dfrac{\dfrac{\dfrac{\dfrac{B \lor C \qquad \neg C \lor \neg D}{B \lor \neg D} \qquad \neg B \lor \neg D}{\neg D} \qquad A \lor D}{A} \qquad \neg A}{\emptyset}$$

Examples

Is $\neg A \land (B \lor C) \land (\neg C \lor \neg D) \land (\neg B \lor \neg D) \land (A \lor D)$ satisfiable?

$$\dfrac{\dfrac{\dfrac{\dfrac{B \lor C \qquad \neg C \lor \neg D}{B \lor \neg D} \qquad \neg B \lor \neg D}{\neg D} \qquad A \lor D}{A} \qquad \neg A}{\emptyset}$$

Alternatively:

$$\dfrac{\dfrac{\neg A \qquad A \lor D}{D} \qquad \dfrac{\dfrac{B \lor C \qquad \neg C \lor \neg D}{B \lor \neg D} \qquad \neg B \lor \neg D}{\neg D}}{\emptyset}$$

Examples

Does $p, (p \wedge q) \Rightarrow r, (s \vee t) \Rightarrow q, t$ entail r?

Examples

Does $p, (p \wedge q) \Rightarrow r, (s \vee t) \Rightarrow q, t$ entail r?

- CNF: $p \wedge (\neg p \vee \neg q \vee r) \wedge (\neg s \vee q) \wedge (\neg t \vee q) \wedge t \wedge \neg r$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\neg p \vee \neg q \vee r \qquad \neg r}{\neg p \vee \neg q} \qquad p}{\neg q} \qquad \neg t \vee q}{\neg t} \qquad t}{\emptyset}
$$

Example

- Coyote chases Roadrunner
- If Roadrunner is smart, Coyote does not catch it
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed
- Roadrunner is smart
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner
  Chase
- If Roadrunner is smart, Coyote does not catch it
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed
- Roadrunner is smart
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner
  $Chase$
- If Roadrunner is smart, Coyote does not catch it
  $Smart \Rightarrow \neg\, Catch$
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed
- Roadrunner is smart
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner

  Chase
- If Roadrunner is smart, Coyote does not catch it

  $Smart \Rightarrow \neg\ Catch$
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed

  $Chase \wedge \neg\ Catch \Rightarrow Annoyed$
- Roadrunner is smart
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner
  $Chase$
- If Roadrunner is smart, Coyote does not catch it
  $Smart \Rightarrow \neg\ Catch$
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed
  $Chase \wedge \neg\ Catch \Rightarrow Annoyed$
- Roadrunner is smart
  $Smart$
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner

  $Chase$
- If Roadrunner is smart, Coyote does not catch it

  $\neg Smart \vee \neg Catch$
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed

  $Chase \wedge \neg Catch \Rightarrow Annoyed$
- Roadrunner is smart

  $Smart$
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner
  Chase
- If Roadrunner is smart, Coyote does not catch it
  $\neg \mathit{Smart} \vee \neg \mathit{Catch}$
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed
  $\neg \mathit{Chase} \vee \mathit{Catch} \vee \mathit{Annoyed}$
- Roadrunner is smart
  Smart
- Question: Is Coyote annoyed?

Example

- Coyote chases Roadrunner

  Chase
- If Roadrunner is smart, Coyote does not catch it

  $\neg\ Smart \vee \neg\ Catch$
- If Coyote chases Roadrunner and does not catch it, then Coyote is annoyed

  $\neg\ Chase \vee Catch \vee Annoyed$
- Roadrunner is smart

  Smart
- Question: Is Coyote annoyed?

  $\neg\ Annoyed$

Example

- $Chase$
- $\neg Smart \vee \neg Catch$
- $\neg Chase \vee Catch \vee Annoyed$
- $Smart$
- $\neg Annoyed$

Example

- $Chase$
- $\neg\ Smart \lor \neg\ Catch$
- $\neg\ Chase \lor\ Catch \lor\ Annoyed$
- $Smart$
- $\neg\ Annoyed$

$$\frac{\neg\ Ann \qquad \neg\ Cha \lor Cat \lor Ann}{\neg\ Cha \lor Cat}$$

Example

- $Chase$
- $\neg\, Smart \vee \neg\, Catch$
- $\neg\, Chase \vee Catch \vee Annoyed$
- $Smart$
- $\neg\, Annoyed$

$$\dfrac{\dfrac{\neg\, Ann \qquad \neg\, Cha \vee Cat \vee Ann}{\neg\, Cha \vee Cat} \qquad Cha}{Cat}$$

Example

- $Chase$
- $\neg\ Smart \vee \neg\ Catch$
- $\neg\ Chase \vee Catch \vee Annoyed$
- $Smart$
- $\neg\ Annoyed$

$$\dfrac{\dfrac{\dfrac{\neg\ Ann \qquad \neg\ Cha \vee Cat \vee Ann}{\neg\ Cha \vee Cat} \qquad Cha}{Cat} \qquad \neg\ Sma \vee \neg\ Cat}{\neg\ Sma}$$

Example

- *Chase*
- $\neg$ *Smart* $\vee \neg$ *Catch*
- $\neg$ *Chase* $\vee$ *Catch* $\vee$ *Annoyed*
- *Smart*
- $\neg$ *Annoyed*

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg\, Ann \qquad \neg\, Cha \vee Cat \vee Ann}{\neg\, Cha \vee Cat} \qquad Cha}{Cat} \qquad \neg\, Sma \vee \neg\, Cat}{\neg\, Sma} \qquad Sma}{\emptyset}$$

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?
  - Yes, with an implication!

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?
  - Yes, with an implication!
- A **Horn clause** is $\boxed{?}$

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?
  - Yes, with an implication!
- A **Horn clause** is $\boxed{?}$ a disjunction of literals where at most one is positive

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?
  - Yes, with an implication!
- A **Horn clause** is $\boxed{?}$ a disjunction of literals where at most one is positive
  - $\boxed{\text{QUESTION}}$ Can also Horn clauses always be rewritten as implications?

Horn clauses

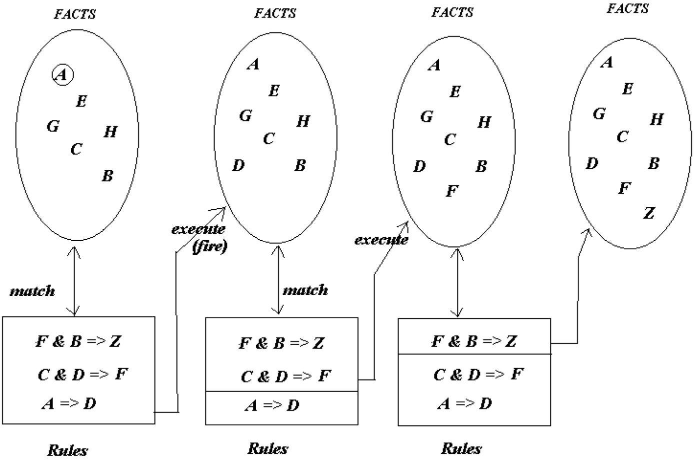
- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?
  - Yes, with an implication!
- A **Horn clause** is $\boxed{?}$ a disjunction of literals where at most one is positive
  - $\boxed{\text{QUESTION}}$ Can also Horn clauses always be rewritten as implications?
  - Yes! E.g. $\neg P \vee \neg Q$ is equivalent to $(P \wedge Q) \Rightarrow \mathit{False}$

Horn clauses

- A **definite clause** is a disjunction of literals where exactly one is positive
  - e.g. $\neg A \vee \neg B \vee C$
  - $\boxed{\text{QUESTION}}$ Can we rewrite it with another connective?
  - Yes, with an implication!
- A **Horn clause** is $\boxed{?}$ a disjunction of literals where at most one is positive
  - $\boxed{\text{QUESTION}}$ Can also Horn clauses always be rewritten as implications?
  - Yes! E.g. $\neg P \vee \neg Q$ is equivalent to $(P \wedge Q) \Rightarrow \mathit{False}$
- Deciding entailment with Horn clauses is easy: **Forward Chaining**

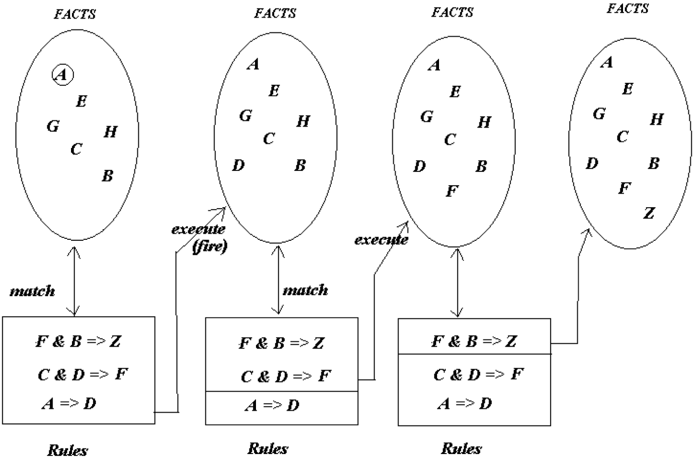
Forward chaining as a graph

Forward chaining (FC) uses modus ponens to process implications



Forward chaining as a graph

Forward chaining (FC) uses modus ponens to process implications



NOTE The set of facts built via FC is actually a model of the KB on which we are applying FC!

Forward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

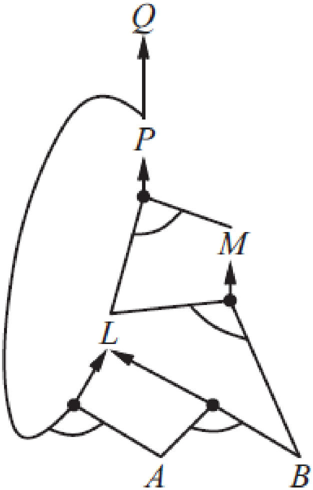
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

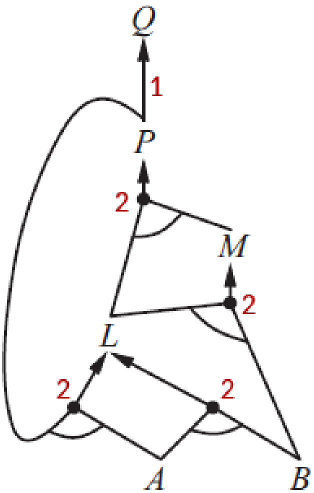
B



The **AND-OR graph** represents the KB in graph form

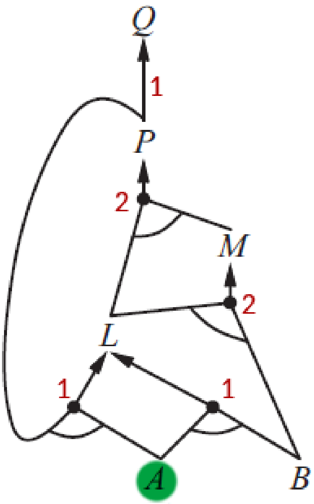
Forward chaining

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



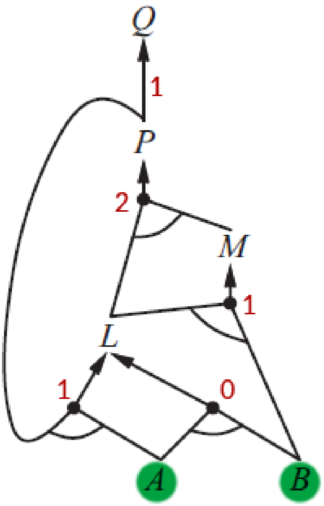
Forward chaining

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



Forward chaining

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



Forward chaining

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

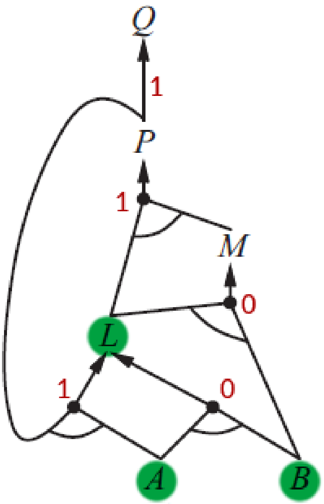
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

$$\cancel{A \wedge B \Rightarrow} \; L$$

$$A$$

$$B$$



Forward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

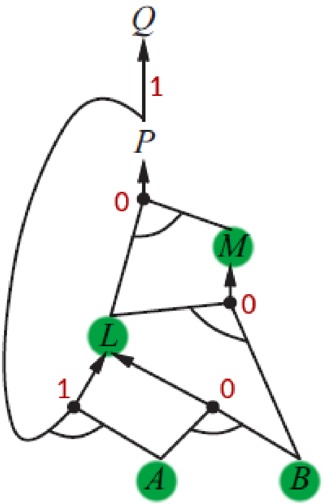
$\cancel{B \wedge L \Rightarrow} M$

$A \wedge P \Rightarrow L$

$\cancel{A \wedge B \Rightarrow} L$

A

B



Forward chaining

$P \Rightarrow Q$

$\cancel{L \wedge M \Rightarrow}\ P$

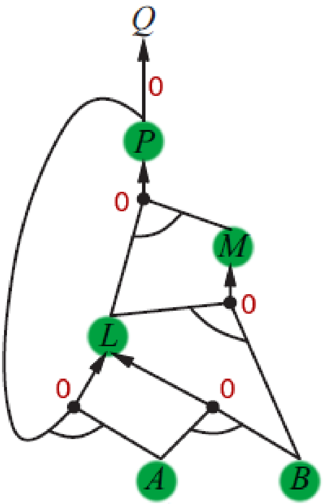
$\cancel{B \wedge L \Rightarrow}\ M$

$A \wedge P \Rightarrow L$

$\cancel{A \wedge B \Rightarrow}\ L$

A

B



Forward chaining

$P \Rightarrow Q$

$\cancel{L \wedge M \Rightarrow} P$

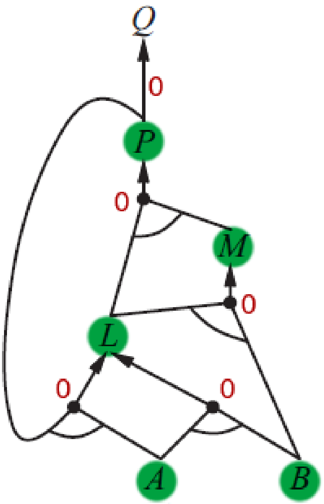
$\cancel{B \wedge L \Rightarrow} M$

$\cancel{A \wedge P \Rightarrow} L$

$\cancel{A \wedge B \Rightarrow} L$

A

B



Forward chaining

$\cancel{P \Rightarrow} Q$

$\cancel{L \wedge M \Rightarrow} P$

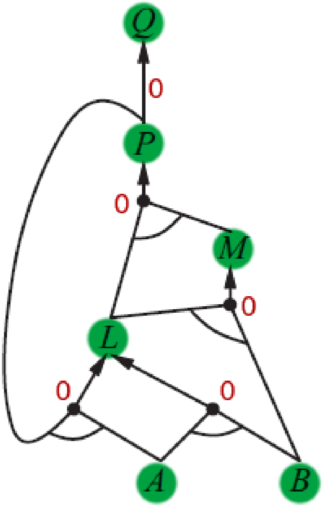
$\cancel{B \wedge L \Rightarrow} M$

$\cancel{A \wedge P \Rightarrow} L$

$\cancel{A \wedge B \Rightarrow} L$

A

B



Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true/false
    inputs: KB, set of definite clauses
            q, the query, a proposition symbol
    count ← a table, of the number of symbols in the premises
    inferred ← a table, initially false for all symbols
    agenda ← a queue, initially true symbols in KB

    while agenda is not empty do
        p ← POP (agenda)
        if p = q then return true
        if inferred[p] = false then
            inferred[p] ← true
            for each clause c in KB where p in c.PREMISE do
                decrement count [c]
                if count [c] = 0 then agenda.ADD(c.CONC)
    return false
```

Forward chaining: soundness and completeness

Soundness

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness Proof towards a contradiction

1. Assume forward chaining has reached a fixed point, where no new atomic sentence can be derived (the *agenda* is empty).

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness Proof towards a contradiction

1. Assume forward chaining has reached a fixed point, where no new atomic sentence can be derived (the *agenda* is empty). Call this state m (it is the model encoded in the table *inferred* of the algorithm).

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness Proof towards a contradiction

1. Assume forward chaining has reached a fixed point, where no new atomic sentence can be derived (the *agenda* is empty). Call this state m (it is the model encoded in the table *inferred* of the algorithm).
2. Suppose there is an atomic sentence b entailed by KB which has not been derived at m

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness Proof towards a contradiction

1. Assume forward chaining has reached a fixed point, where no new atomic sentence can be derived (the *agenda* is empty). Call this state m (it is the model encoded in the table *inferred* of the algorithm).
2. Suppose there is an atomic sentence b entailed by KB which has not been derived at m
   - So there is a clause $a_1 \wedge \cdots \wedge a_k \Rightarrow b$ in KB but b has not been derived at m

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness Proof towards a contradiction

1. Assume forward chaining has reached a fixed point, where no new atomic sentence can be derived (the *agenda* is empty). Call this state m (it is the model encoded in the table *inferred* of the algorithm).
2. Suppose there is an atomic sentence b entailed by KB which has not been derived at m
   - So there is a clause $a_1 \wedge \cdots \wedge a_k \Rightarrow b$ in KB but b has not been derived at m
   - But at m everything that could be derived has been derived (by assumption Step 1)

Forward chaining: soundness and completeness

Soundness

- Since forward chaining only uses modus ponens, and modus ponens only derives true sentences from true sentences, FC is therefore sound

Completeness Proof towards a contradiction

1. Assume forward chaining has reached a fixed point, where no new atomic sentence can be derived (the *agenda* is empty). Call this state m (it is the model encoded in the table *inferred* of the algorithm).
2. Suppose there is an atomic sentence b entailed by KB which has not been derived at m
   - So there is a clause $a_1 \wedge \cdots \wedge a_k \Rightarrow b$ in KB but b has not been derived at m
   - But at m everything that could be derived has been derived (by assumption Step 1)
   - Contradiction
3. Therefore, forward chaining is complete

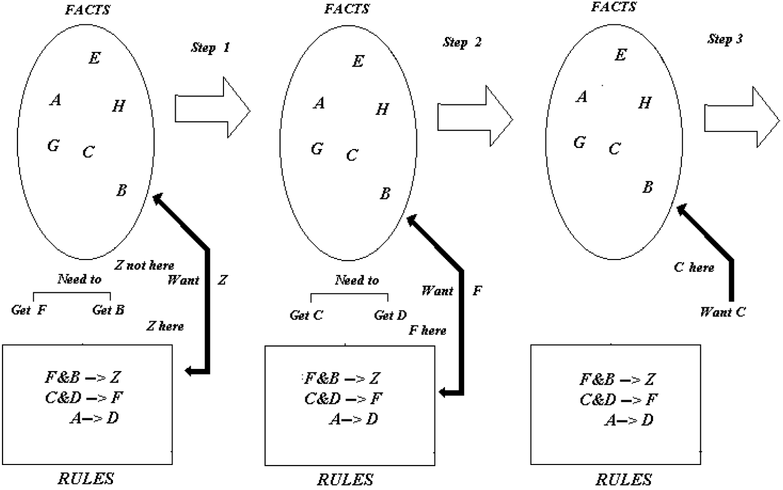
Backward chaining

Backward chaining works backwards from a query q

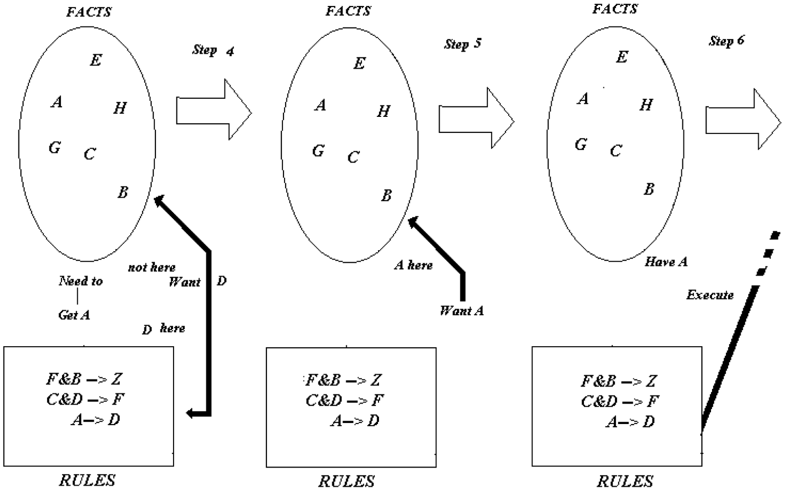
- Finds implications in KB whose conclusion is q
- Prove the premises of one of those implications by backward chaining

- Goal-directed: it does not trigger rules that are not needed to prove the conclusion

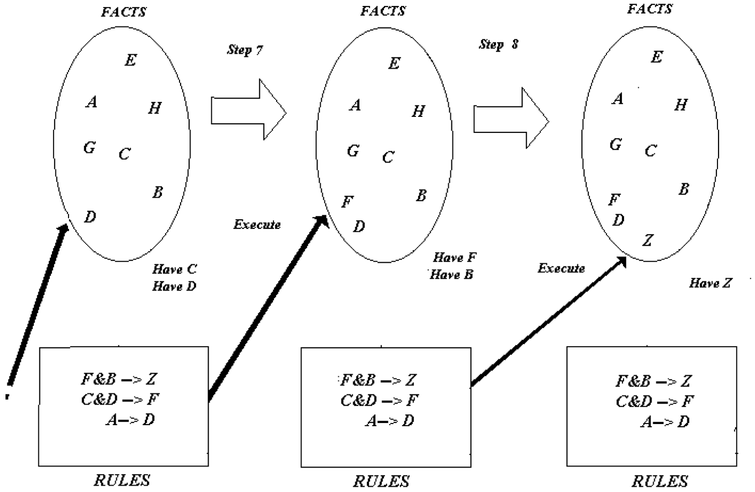
Backward chaining



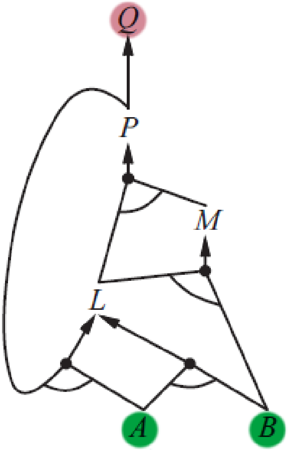
Backward chaining



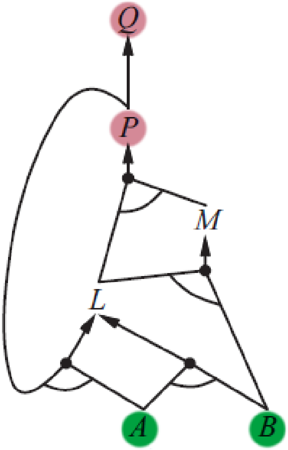
Backward chaining



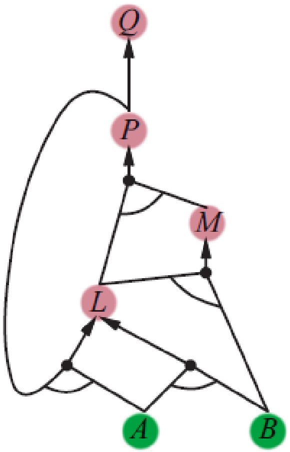
Backward chaining



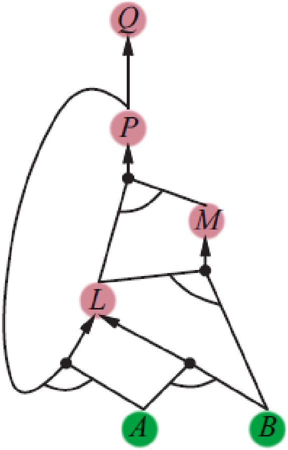
Backward chaining



Backward chaining

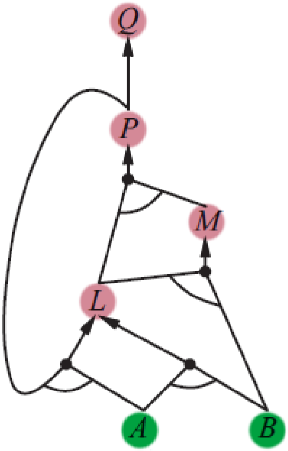


Backward chaining



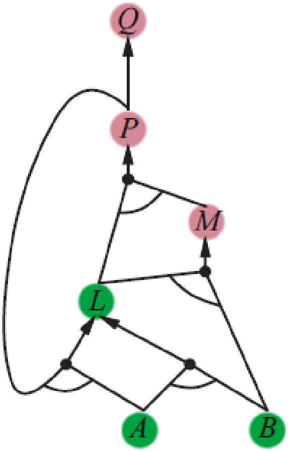
M is proven by L and B, but L is on the goal stack

Backward chaining



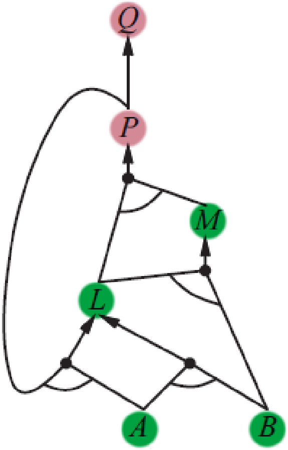
L could be proven by P and A, but P is on the goal stack

Backward chaining

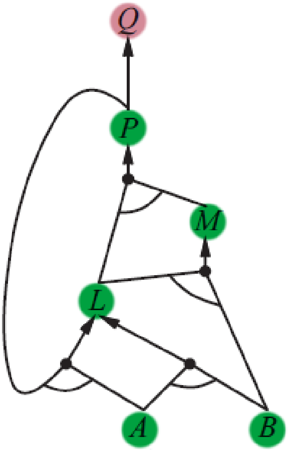


L is proven by A and B, which are known to be true

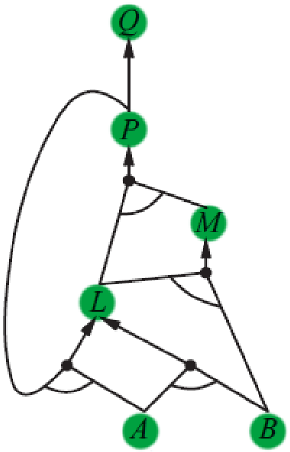
Backward chaining



Backward chaining



Backward chaining



Forward vs backward chaining

- Forward chaining is data-driven
  - Automatic, unconscious processing
  - May conclude many symbols that are irrelevant to the goal
- Backward chaining goal-driven
  - Appropriate for problem-solving
- Actual number of executed actions by backward chaining can be much less than linear in size of KB!

Summary

- Sat Solving
- Theorem proving
- Resolution
- Definite and Horn clauses
- Forward & Backward chaining