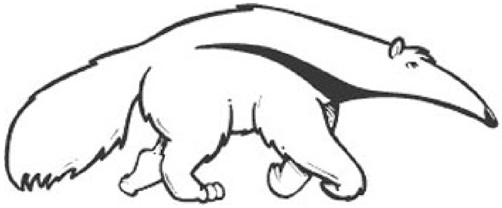
Propositional Logic B: Inference, Reasoning, Proof

CS171, Winter Quarter, 2019
Introduction to Artificial Intelligence
Prof. Richard Lathrop



Read Beforehand: **R&N 7.1-7.5 (optional: 7.6-7.8)**

BREN:ICS
INFORMATION AND COMPUTER SCIENCES
UNIVERSITY of CALIFORNIA IRVINE

You will be expected to know

- Basic definitions
  - Inference, derive, sound, complete
- Conjunctive Normal Form (CNF)
  - Convert a Boolean formula to CNF
- Do a short resolution proof
- Horn Clauses
- Do a short forward-chaining proof
- Do a short backward-chaining proof
- Model checking with backtracking search
- Model checking with local search

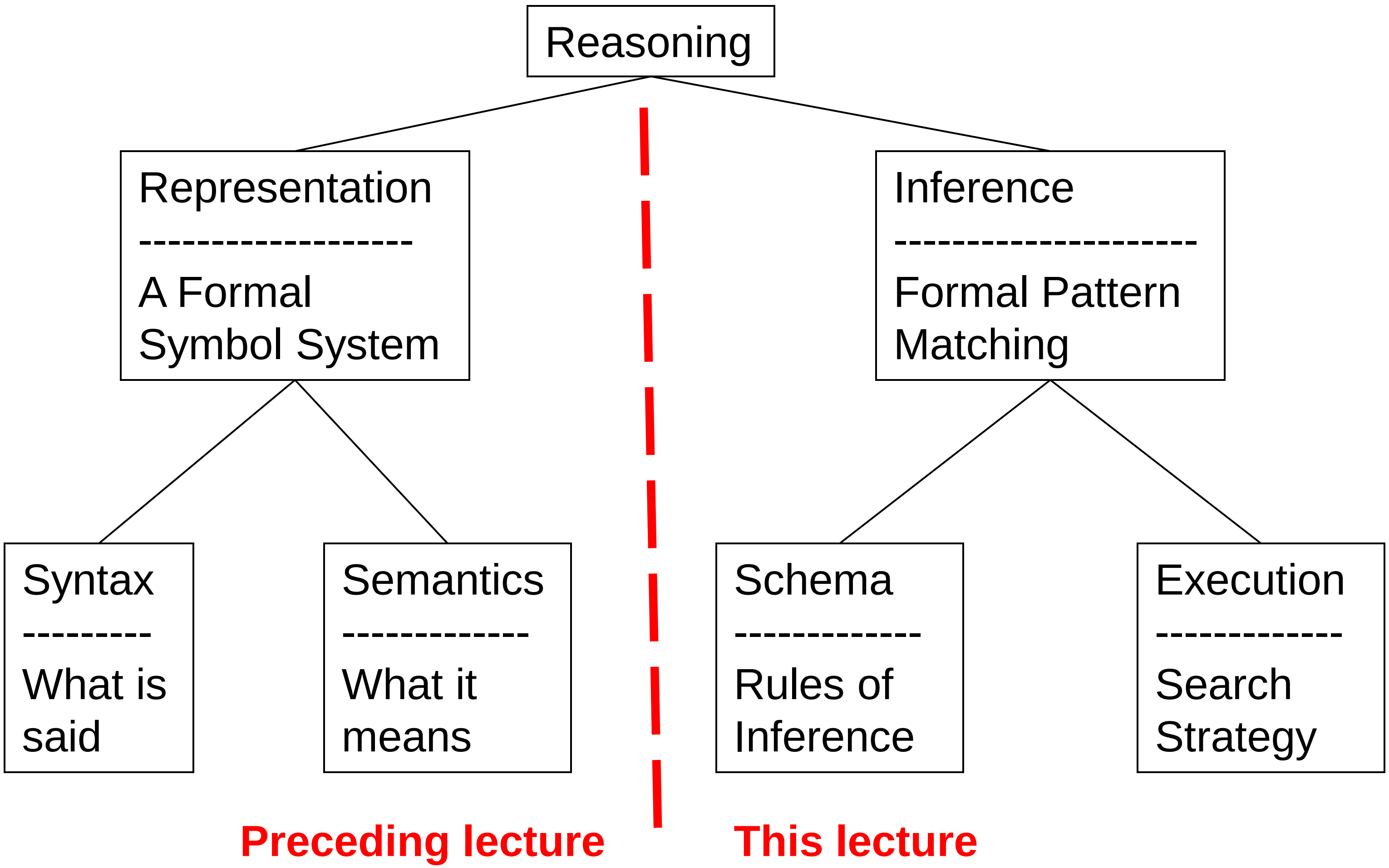
Review: Inference in Formal Symbol Systems Ontology, Representation, Inference

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

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

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

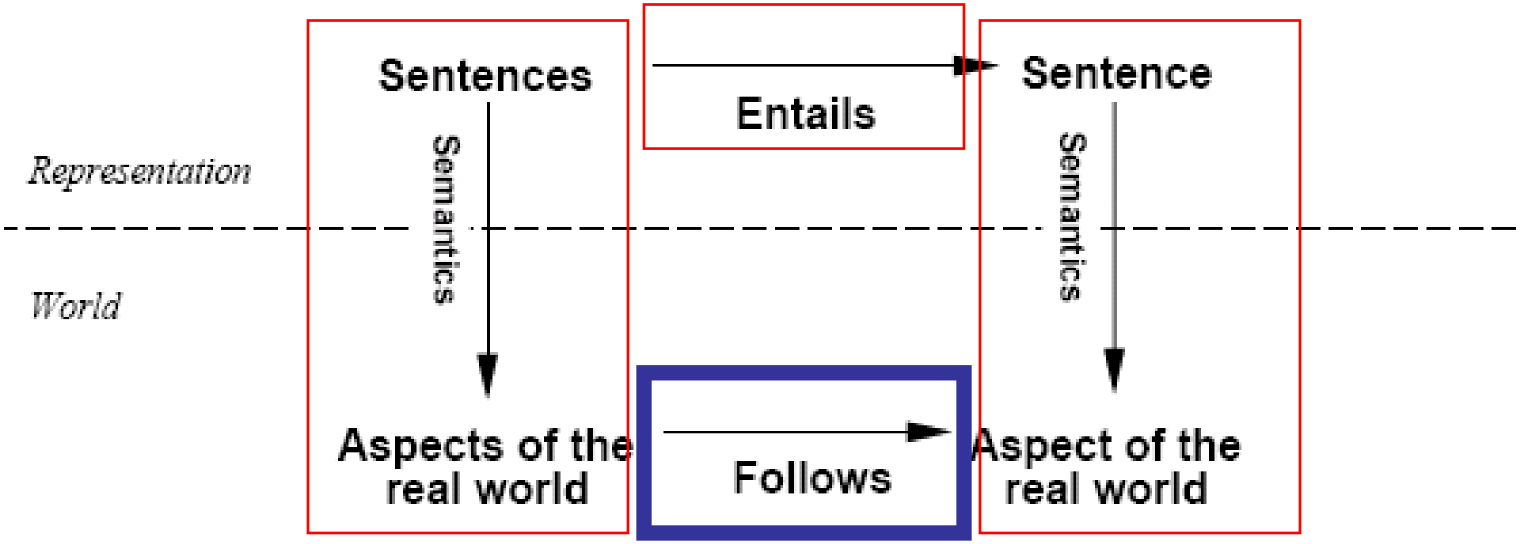
Review



Review

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

Review: Schematic perspective



If KB is true in the real world,
then any sentence α ***entailed*** *by KB*
is also true in the real world.

For example: If I tell you (1) Sue is Mary's sister, and (2) Sue is Amy's mother, then it necessarily follows in the world that Mary is Amy's aunt, even though I told you nothing at all about aunts. This sort of reasoning pattern is what we hope to capture.

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

Is this inference correct?

How do you know?
How can you tell?

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

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

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

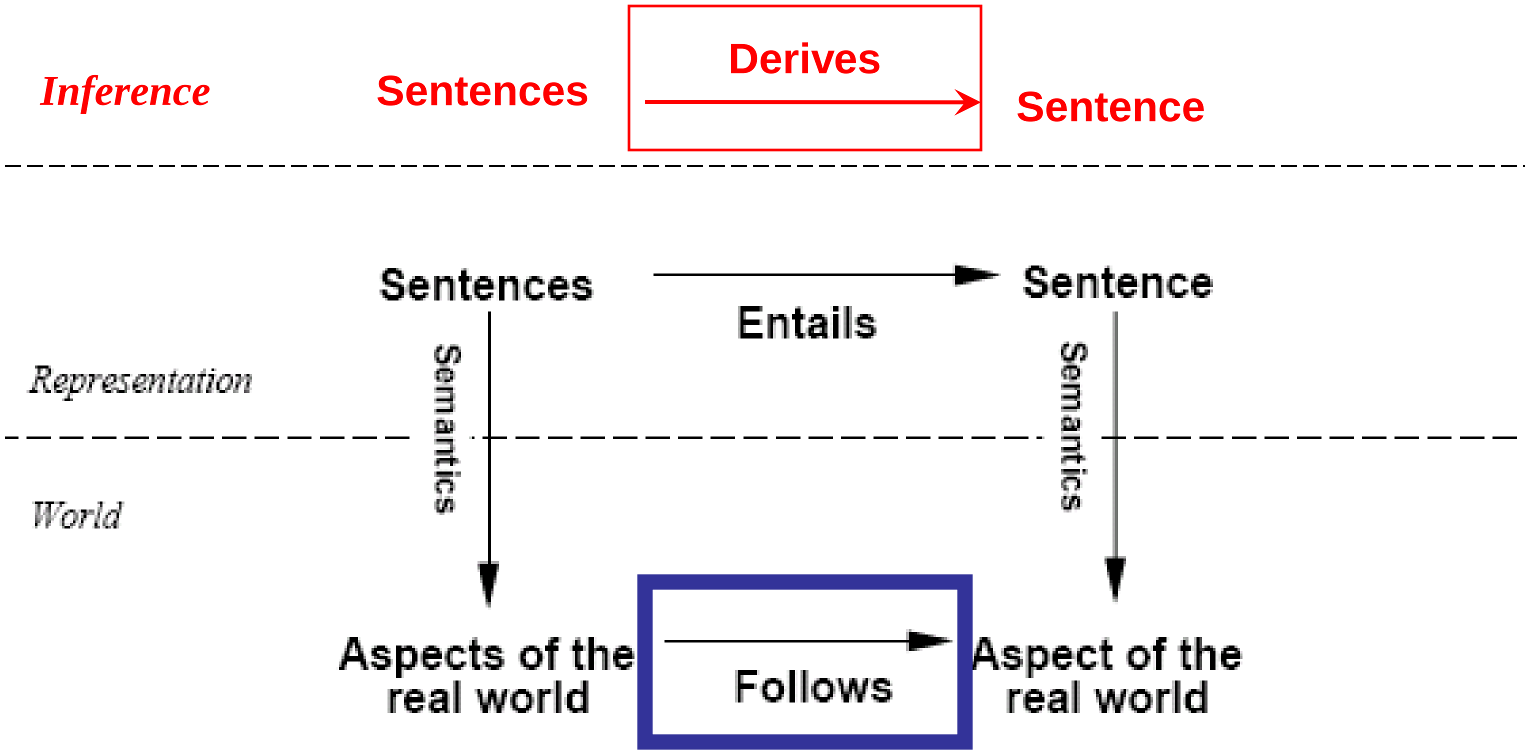
Is this inference correct?

How do you know?
How can you tell?

- All men are people;
  Half of all people are women;
  Therefore, half of all men are women.

- Penguins are black and white;
  Some old TV shows are black and white;
  Therefore, some penguins are old TV shows.

Schematic perspective



If KB is true in the real world,

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

by a sound inference procedure

is also true in the real world.

Logical inference

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

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

- *Sound* (or *truth preserving*):
  The algorithm **only** derives entailed sentences.
  - Otherwise it just makes things up.
    i is sound iff whenever KB $|-_i \alpha$ *it is also true that KB* $|= \alpha$
  - *E.g., model-checking is sound*
  Refusing to infer any sentence is Sound; so, Sound is weak alone.
- *Complete*:
  The algorithm can derive **every** entailed sentence.
    i is complete iff whenever KB $|= \alpha$ *it is also true that KB* $|-_i \alpha$
  Deriving every sentence is Complete; so, Complete is weak alone.

Proof methods

- Proof methods divide into (roughly) two kinds:

  Application of inference rules:

  Legitimate (sound) generation of new sentences from old.

  – Resolution --- KB is in Conjunctive Normal Form (CNF)
  – Forward & Backward chaining

  Model checking:

  Searching through truth assignments.

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

Examples of Sound Inference Patterns

Classical Syllogism (due to Aristotle)

All Ps are Qs	All Men are Mortal
X is a P	Socrates is a Man
Therefore, X is a Q	Therefore, Socrates is Mortal

Implication (Modus Ponens)

P implies Q	Smoke implies Fire
P	Smoke
Therefore, Q	Therefore, Fire

Why is this different from:
All men are people
Half of people are women
So half of men are women

Contrapositive (Modus Tollens)

P implies Q	Smoke implies Fire
Not Q	Not Fire
Therefore, Not P	Therefore, not Smoke

Law of the Excluded Middle (due to Aristotle)

A Or B	Alice is a Democrat or a Republican
Not A	Alice is not a Democrat
Therefore, B	Therefore, Alice is a Republican

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

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
  - DNF = Disjunctive Normal Form
    - A disjunct of conjuncts = (OR (AND …) (AND …))
    - DNF is used by Decision Trees in Machine Learning
- Can convert any Boolean formula to CNF or DNF

Clause

Term

Conjunctive Normal Form (CNF)

We'd like to prove: KB $|=$ α

(This is equivalent to KB $\wedge \neg \alpha$ is unsatisfiable.)

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

A "conjunction of disjunctions"

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

Clause: $(A \vee \neg B)$ — Clause: $(B \vee \neg C \vee \neg D)$

literals: $\neg C$, $\neg D$

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

Review: Equivalence & Implication

- Equivalence is a conjoined double implication
  - $(X \Leftrightarrow Y) = [(X \Rightarrow Y) \wedge (Y \Rightarrow X)]$
- Implication is (NOT antecedent OR consequent)
  - $(X \Rightarrow Y) = (\neg X \vee Y)$

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

- **<u>Both</u>** 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
  - $X \vee (Y \wedge Z) = (X \vee Y) \wedge (X \vee Z)$
  - $(W \wedge X) \vee (Y \wedge Z) = (W \vee Y) \wedge (X \vee Y) \wedge (W \vee Z) \wedge (X \vee Z)$

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

From the previous slide we had:

$= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

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

KB =

...

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

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

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

...

(same)

Often, Won't Write "$\lor$" or "$\land$"

(we know they are there)

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

Resolution = Efficient Implication

Recall that (A => B) = ((NOT A) OR B)
and so:

(Y OR X) = ((NOT X) => Y)

((NOT Y) OR Z) = (Y => Z)

which yields:

((Y OR X) AND ((NOT Y) OR Z)) |= ((NOT X) => Z) = (X OR Z)

(OR A B C D)	->Same ->	(NOT (OR B C D)) => A
(OR ¬A E F G)	->Same ->	A => (OR E F G)
----------------------------		---
(OR B C D E F G)		(NOT (OR B C D)) => (OR E F G)

		(OR B C D E F G)

Recall: All clauses in KB are conjoined by an implicit AND (= CNF representation).

Resolution Examples

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

$(A \vee B \vee C)$

$(\neg A)$ "If A or B or C is true, but not A, then B or C must be true."

— — — — — — — — — — — —

$\therefore (B \vee C)$

$(A \vee B \vee C)$

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

— — — — — — — — — — — —

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

"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."

$(A \vee B)$

$(\neg A \vee B)$

— — — — — — — —

$\therefore (B \vee B) \equiv B$ ← Simplification is done always.

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

* Resolution is "refutation complete" in that it can prove the truth of any entailed sentence by refutation.
* You can start two resolution proofs in parallel, one for the sentence and one for its negation, and see which branch returns a correct proof.

More Resolution Examples

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

Only Resolve <u>ONE</u> Literal Pair!

If more than one pair, result always = TRUE.

<u>Useless!!</u> Always simplifies to TRUE!!

No!

(OR A B C D)
(OR ¬A ¬B F G)

(OR C D F G)

No! This is wrong!

No!

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR D)

No! This is wrong!

Yes! (but = TRUE)

(OR A B C D)
(OR ¬A ¬B F G)

(OR B ¬B C D F G)

Yes! (but = TRUE)

Yes! (but = TRUE)

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR A ¬A B ¬B D)

Yes! (but = TRUE)

(Resolution theorem provers routinely pre-scan the two clauses for two complementary literals, and if they are found won't resolve those clauses.)

Resolution Algorithm

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

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

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

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

* I.e. = *id est* = that is

Resolution example

Stated in English

- "Laws of Physics" in the Wumpus World:
  - "A breeze in B11 is equivalent to a pit in P12 or a pit in P21."
- Particular facts about a specific instance:
  - "There is no breeze in B11."
- Goal or query sentence:
  - "Is it true that P12 does not have a pit?"

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."

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

We converted this sentence to CNF in the CNF example we worked above.

- 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?"

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

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:

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

- <u>Negated</u> 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 ( ):

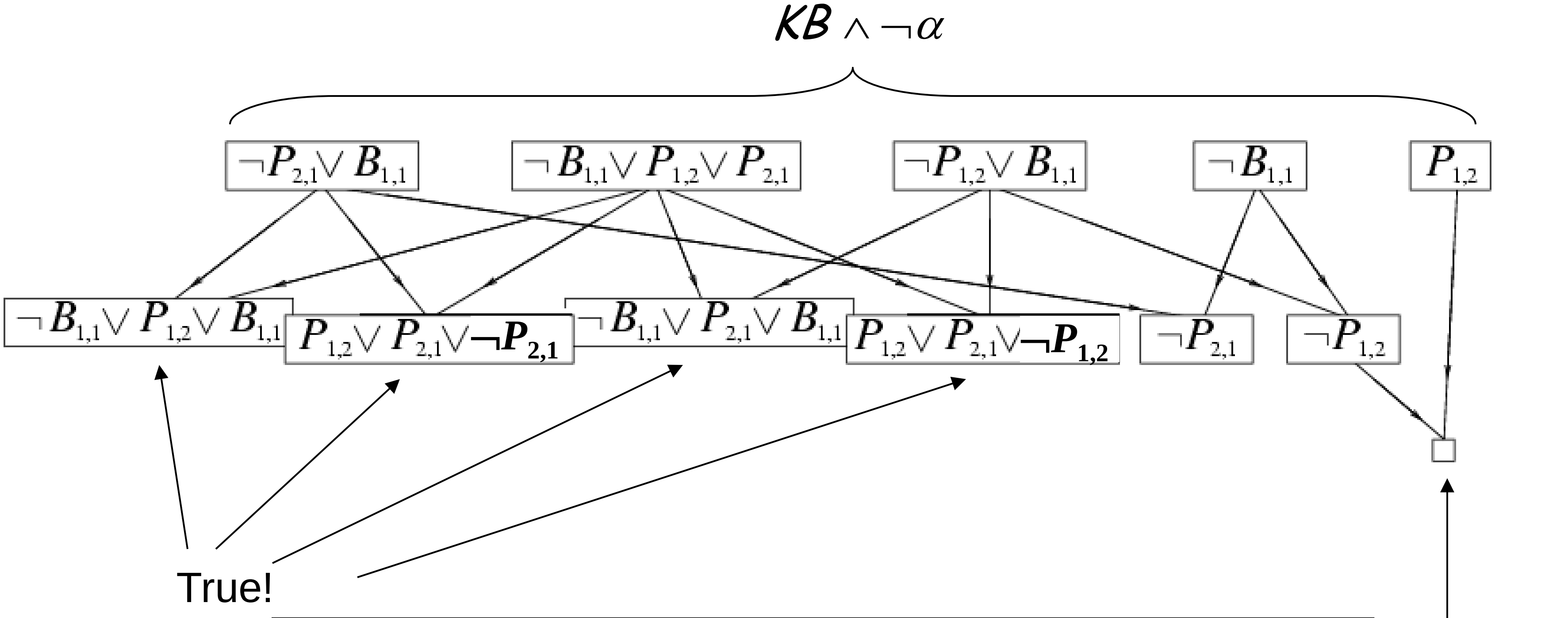
- Resolve $(\neg P_{1,2} \quad B_{1,1})$ and $(\neg B_{1,1})$ to give $(\neg P_{1,2})$
- Resolve $(\neg P_{1,2})$ and $(P_{1,2})$ to give $(\ )$

- 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

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



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.

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  <u>*Prove that the unicorn is both magical and horned.*</u>

Problem 7.2, R&N page 280. (Adapted from Barwise and Etchemendy, 1993.)

Note for non-native-English speakers: immortal = not mortal

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

- **First, Ontology**: What do we need to describe and reason about?

- Use these propositional variables ("immortal" = "not mortal"):

  Y = unicorn is mYthical

  R = unicorn is moRtal

  M = unicorn is a maMmal

  H = unicorn is Horned

  G = unicorn is maGical

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical   R = unicorn is moRtal

  M = unicorn is a maMmal   H = unicorn is Horned

  G = unicorn is maGical

- **Second, translate to Propositional Logic, then to CNF:**
- Propositional logic (prefix form, aka Polish notation):
  - **(=> Y (NOT R))**   ; same as (Y => (NOT R)) in infix form
- CNF (clausal form)   ; recall (A => B) = ((NOT A) OR B)
  - **((NOT Y) (NOT R))**

Prefix form is often a better representation for a parser, since it looks at the first element of the list and dispatches to a handler for that operator token.

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical
  R = unicorn is moRtal
  M = unicorn is a maMmal
  H = unicorn is Horned
  G = unicorn is maGical

- **Second, translate to Propositional Logic, then to CNF**:
- Propositional logic (prefix form):
  - **(=> (NOT Y) (AND R M))** ;same as ((NOT Y) => (R AND M)) in infix form
- CNF (clausal form)
  - **(M Y)**
  - **(R Y)**

If you ever have to do this "for real" you will likely invent a new domain language that allows you to state important properties of the domain --- then parse that into propositional logic, and then CNF.

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical
  R = unicorn is moRtal
  M = unicorn is a maMmal
  H = unicorn is Horned
  G = unicorn is maGical

- **Second, translate to Propositional Logic, then to CNF:**
- Propositional logic (prefix form):
  - **(=> (OR (NOT R) M) H)** ; same as ((Not R) OR M) => H in infix form
- CNF (clausal form)
  - **(H (NOT M))**
  - **(H R)**

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical
  R = unicorn is moRtal
  M = unicorn is a maMmal
  H = unicorn is Horned
  G = unicorn is maGical

- **Second, translate to Propositional Logic, then to CNF**:
- Propositional logic (prefix form)
  - **(=> H G)** ; same as H => G in infix form
- CNF (clausal form)
  - **((NOT H) G)**

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical
  R = unicorn is moRtal
  M = unicorn is a maMmal
  H = unicorn is Horned
  G = unicorn is maGical

- **Current KB** (in CNF clausal form) =

  ((NOT Y) (NOT R))
  (M Y)
  (R Y)
  (H (NOT M))
  (H R)
  ((NOT H) G)

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical

  R = unicorn is moRtal

  M = unicorn is a maMmal

  H = unicorn is Horned

  G = unicorn is maGical

- **Third, negated goal to Propositional Logic, then to CNF:**
- Goal sentence in propositional logic (prefix form)
  - **(AND H G)** ; same as H AND G in infix form
- Negated goal sentence in propositional logic (prefix form)
  - **(NOT (AND H G)) = (OR (NOT H) (NOT G))**
- CNF (clausal form)
  - **((NOT G) (NOT H))**

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

  Y = unicorn is mYthical R = unicorn is moRtal

  M = unicorn is a maMmal H = unicorn is Horned

  G = unicorn is maGical

- **Current KB + negated goal** (in CNF clausal form) =

((NOT Y) (NOT R)) **(M Y)** **(R Y)** **(H (NOT M))**

(H R) **((NOT H) G)** **((NOT G) (NOT H))**

Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

  Prove that the unicorn is both magical and horned.

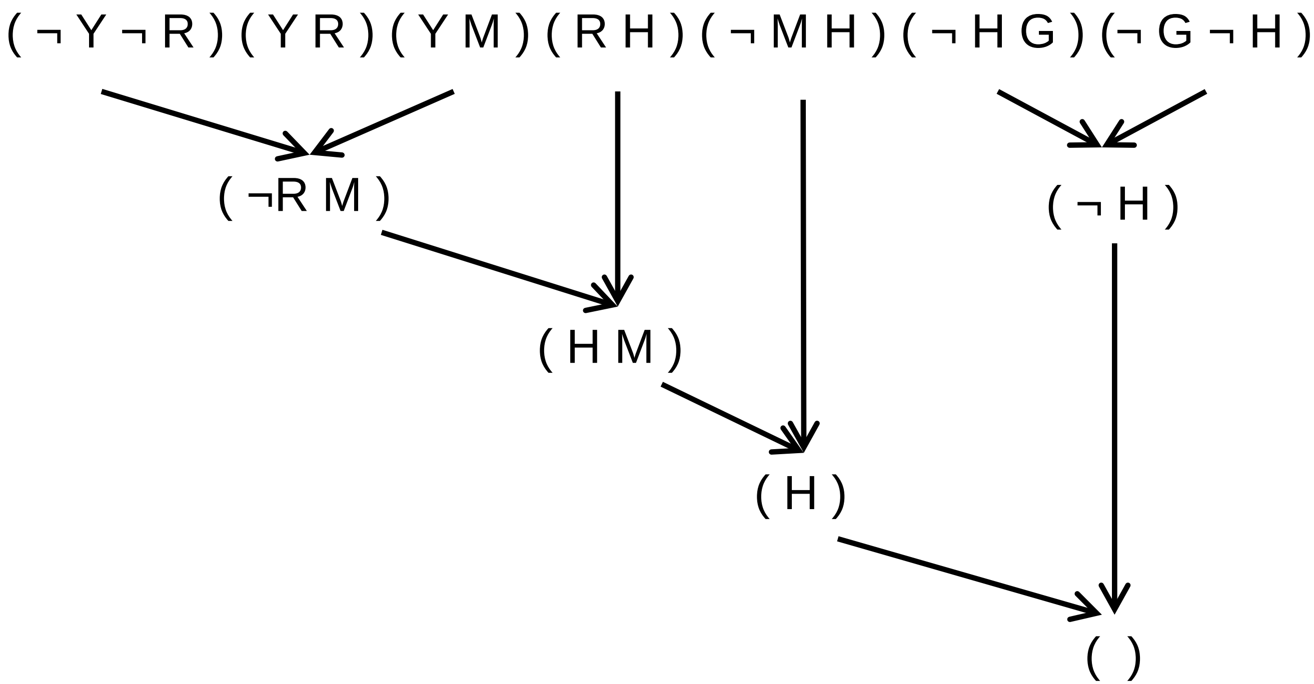
((NOT Y) (NOT R))	**(M Y)**	**(R Y)**	**(H (NOT M))**
(H R)	**((NOT H) G)**	**((NOT G) (NOT H))**	

- **Fourth, produce a resolution proof ending in ( ):**
- Resolve (¬H ¬G) and (¬H G) to give (¬H)
- Resolve (¬Y ¬R) and (Y M) to give (¬R M)
- Resolve (¬R M) and (R H) to give (M H)
- Resolve (M H) and (¬M H) to give (H)
- Resolve (¬H) and (H) to give ( )

- Of course, there are many other proofs, which are OK iff correct.

Detailed Resolution Proof Example

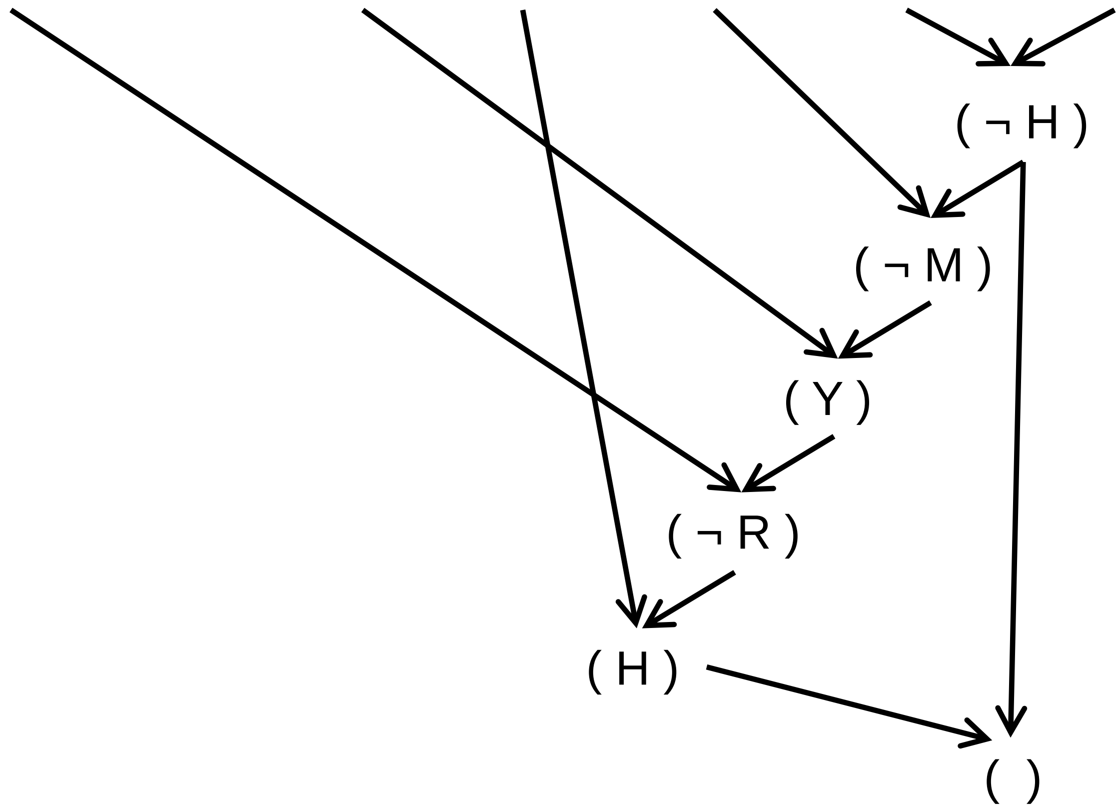
Graph view of proof



Detailed Resolution Proof Example

Graph view of a different proof

- (¬ Y ¬ R) (Y R) (Y M) (R H) (¬ M H) (¬ H G) (¬ G ¬ H)



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 ≥ 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 ∧ B) must be false
- 0 negative literals: fact

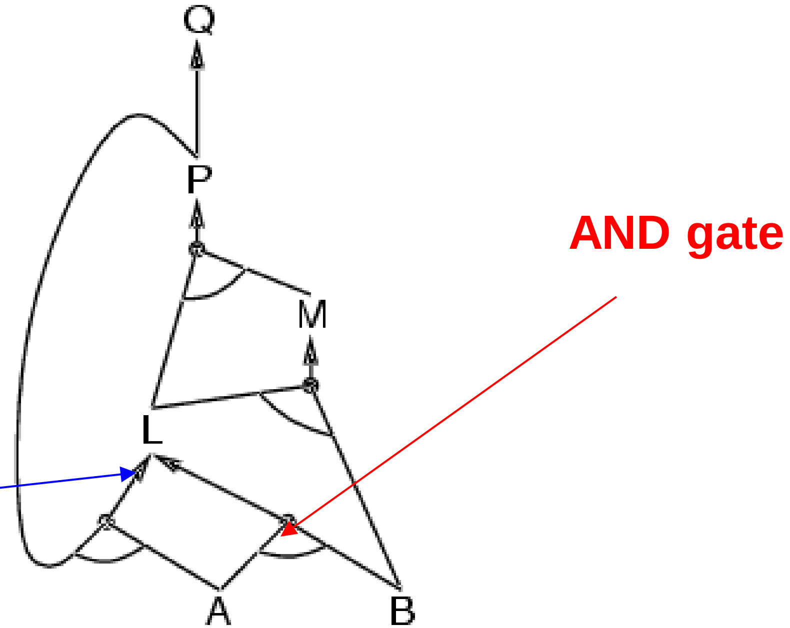
  e.g., (A) ≡ (True ⇒ A) states that A must be true.
- Forward Chaining and Backward chaining are sound and complete with Horn clauses and run linear in space and time.

Forward chaining (FC)

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

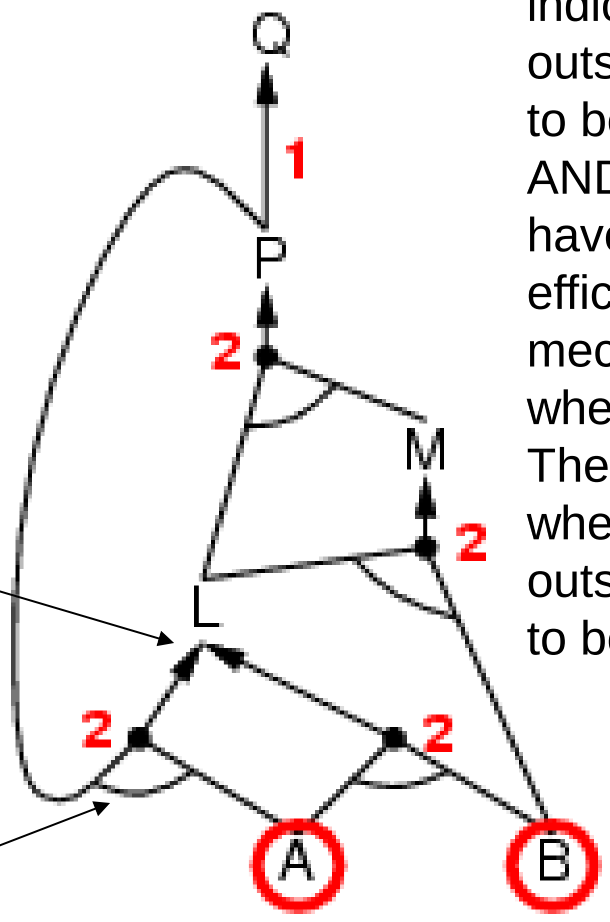
- This proves that *KB* $\Rightarrow$ *Query* is true in all possible worlds (i.e. trivial), and hence it proves entailment.

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



- Forward chaining is sound and complete for Horn KB

Forward chaining example

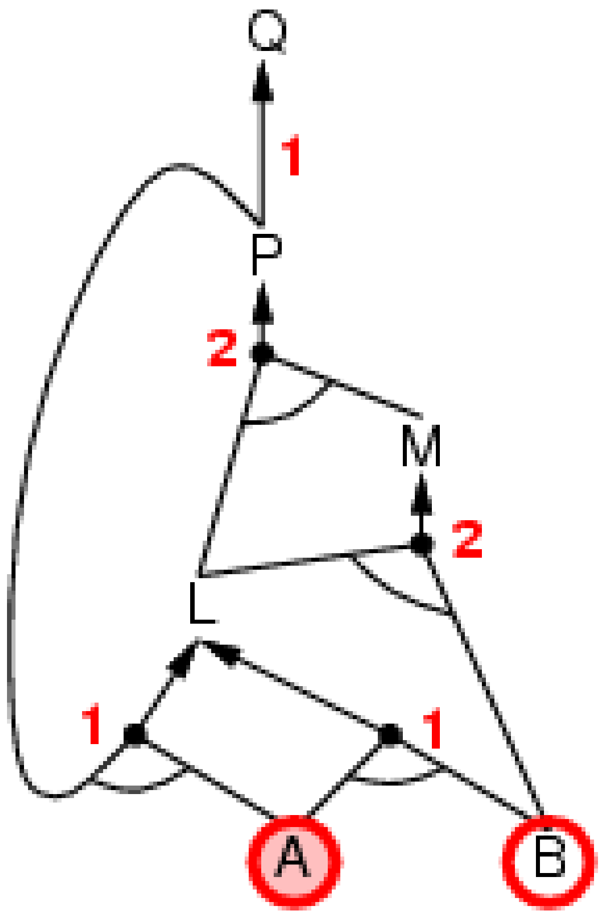


"OR" Gate

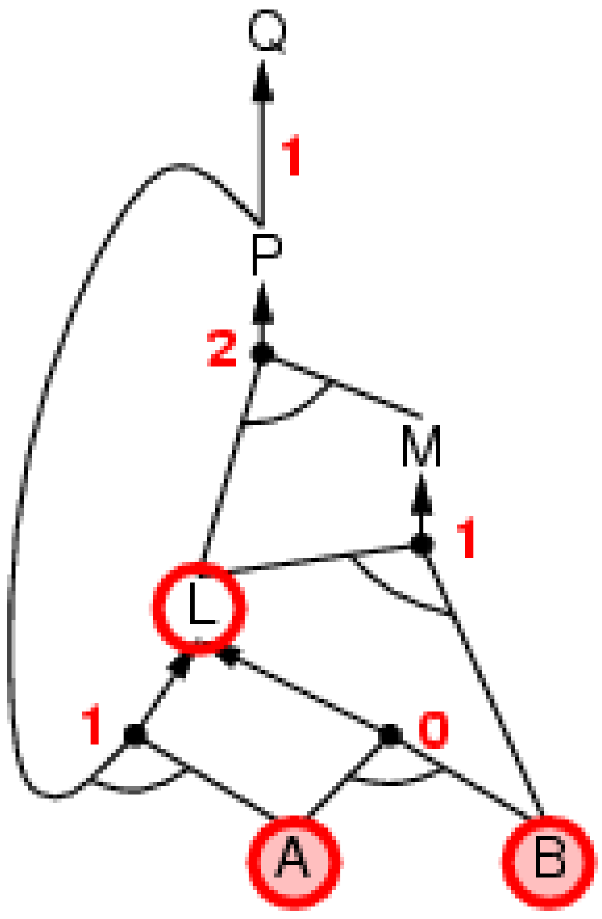
"AND" gate

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.

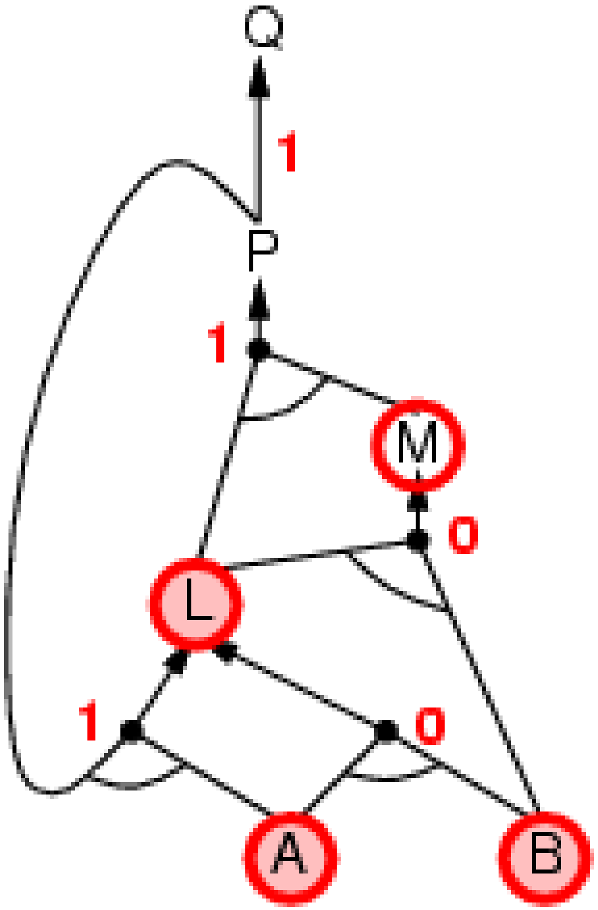
Forward chaining example



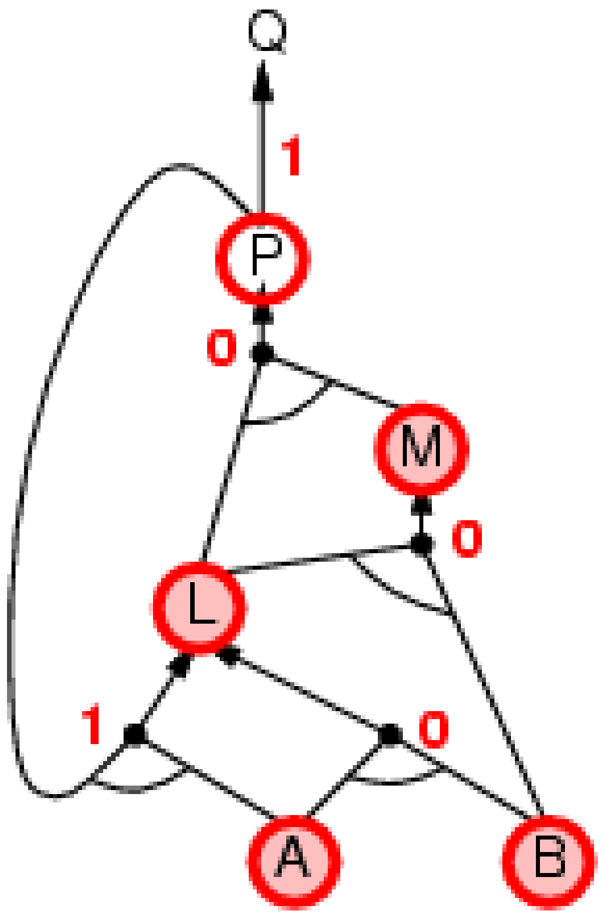
Forward chaining example



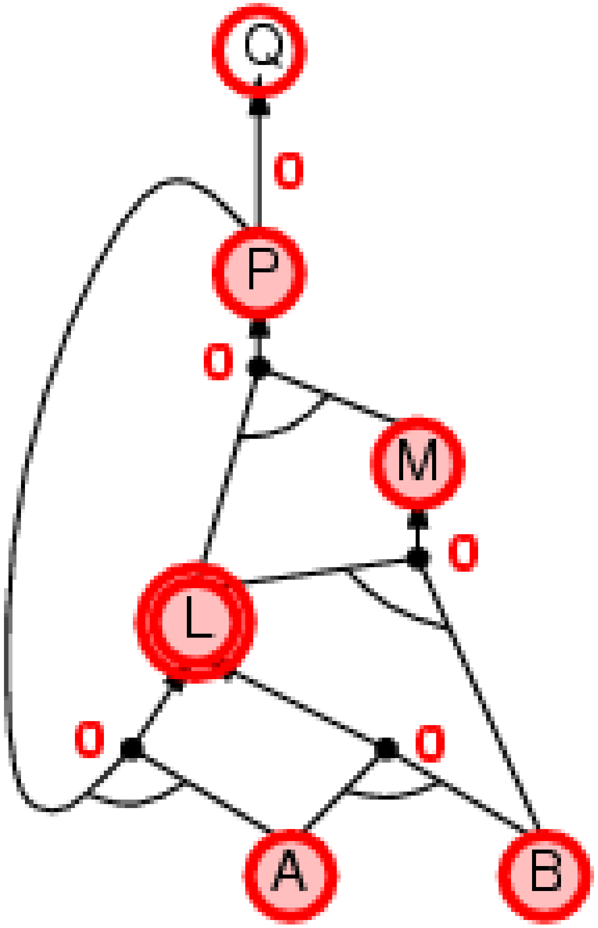
Forward chaining example



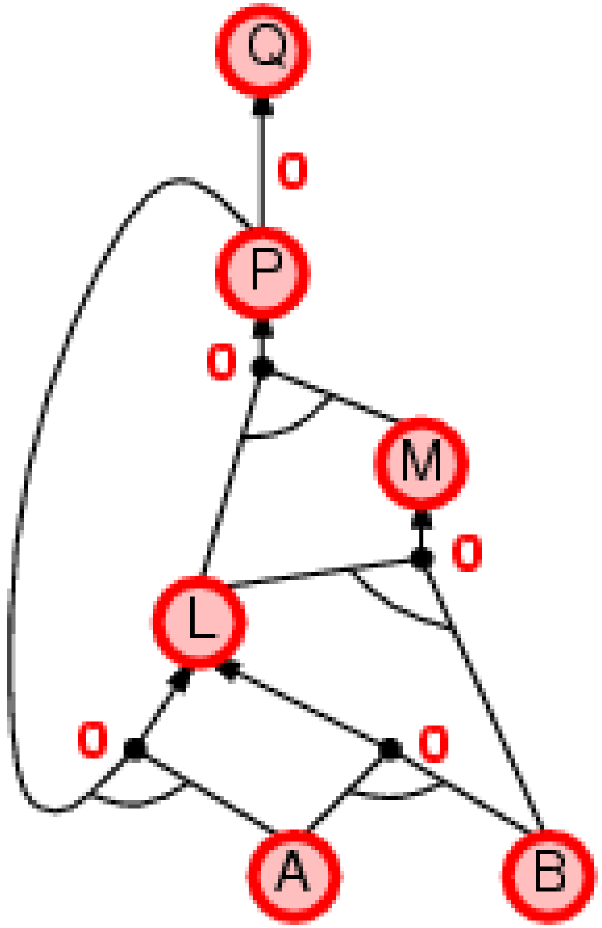
Forward chaining example



Forward chaining example



Forward chaining example



Backward chaining (BC)

Idea: work backwards from the query *q*

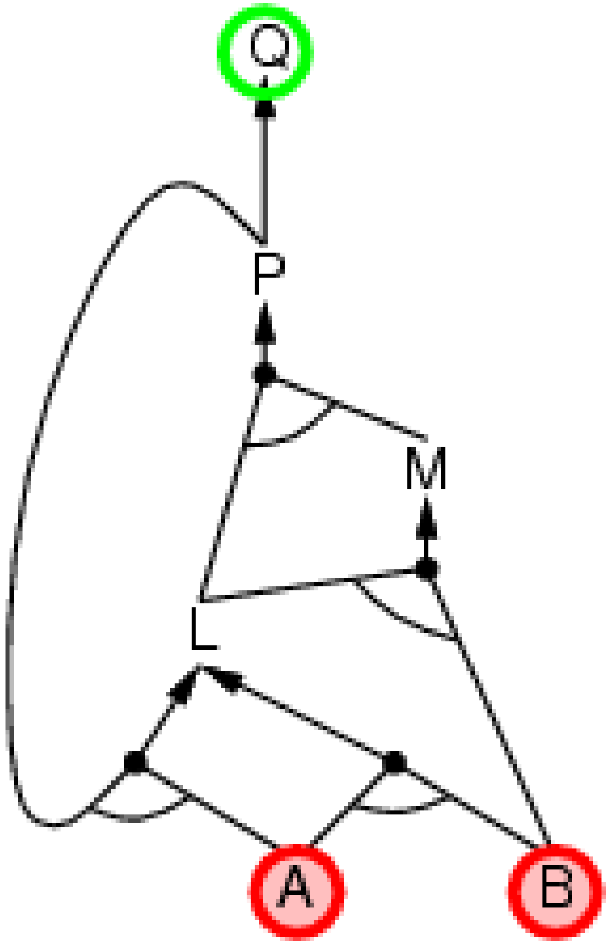
- check if *q* is known already, or
- prove by BC all premises of some rule concluding *q*
- Hence BC maintains a stack of sub-goals that need to be proved to get to q.

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

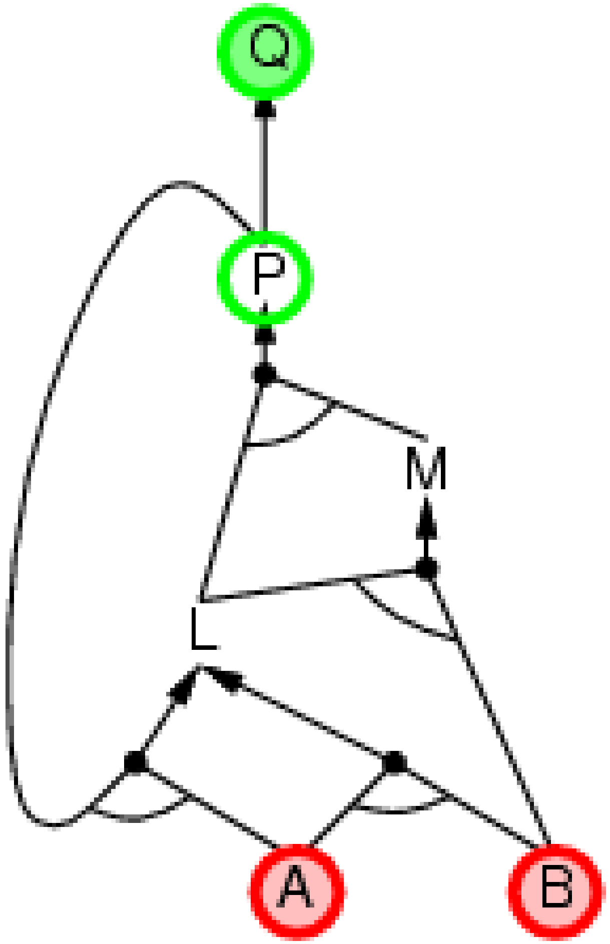
Avoid repeated work: check if new sub-goal

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

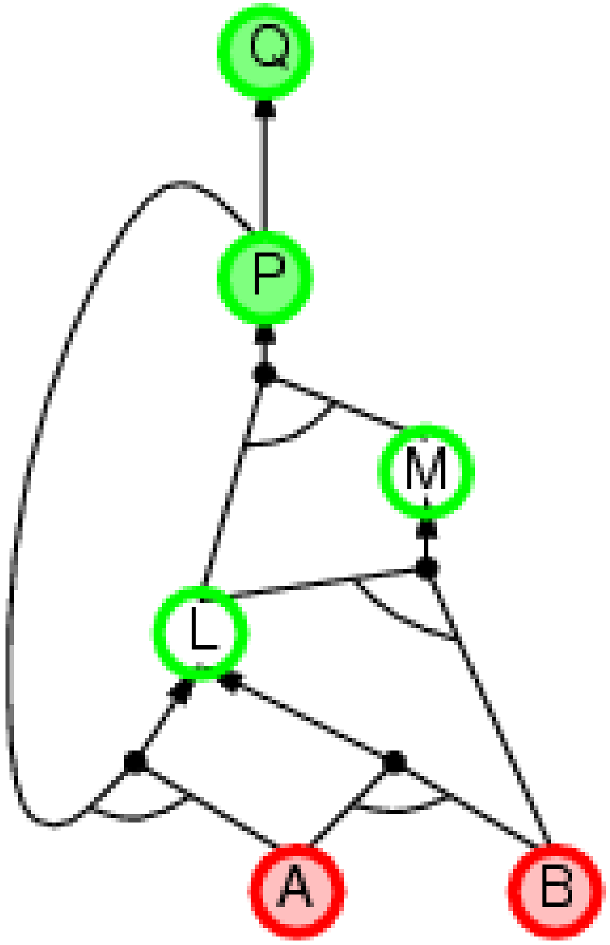
Backward chaining example



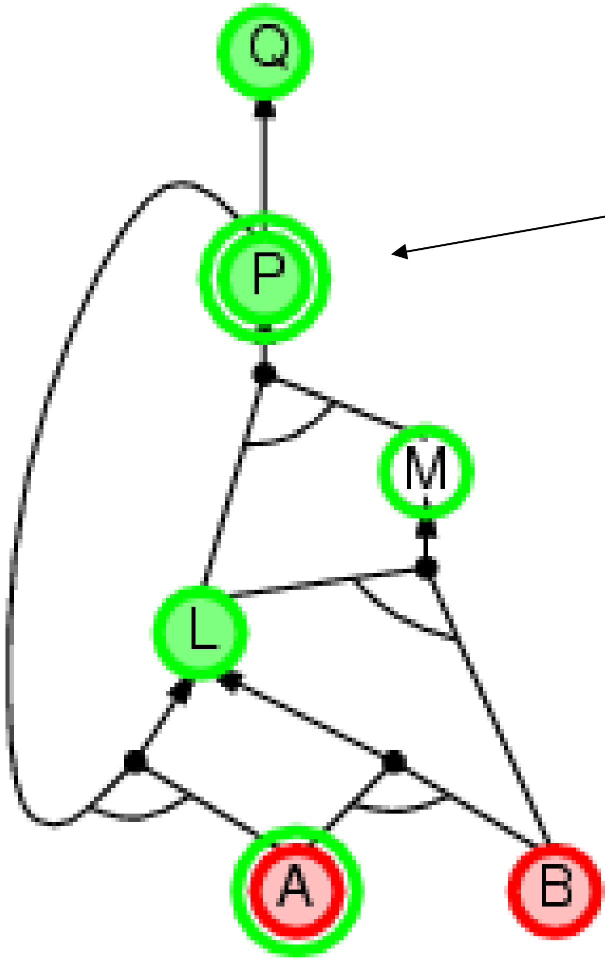
Backward chaining example



Backward chaining example

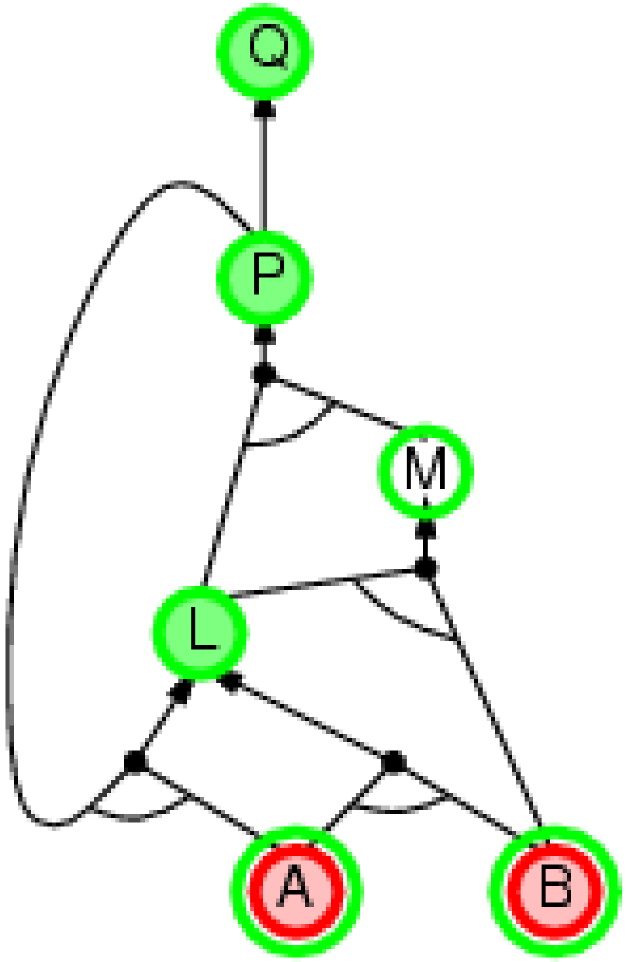


Backward chaining example



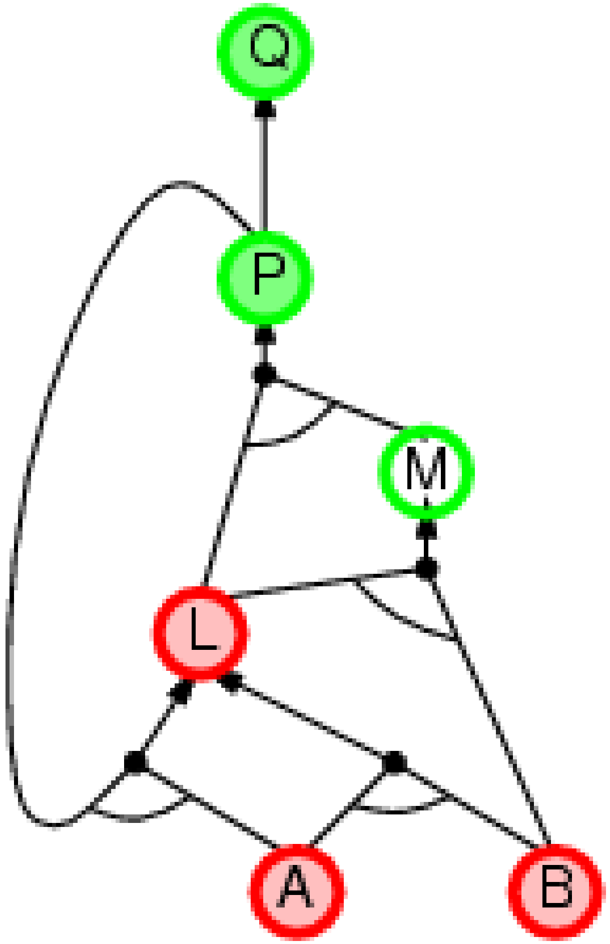
we need P to prove L and L to prove P.

Backward chaining example

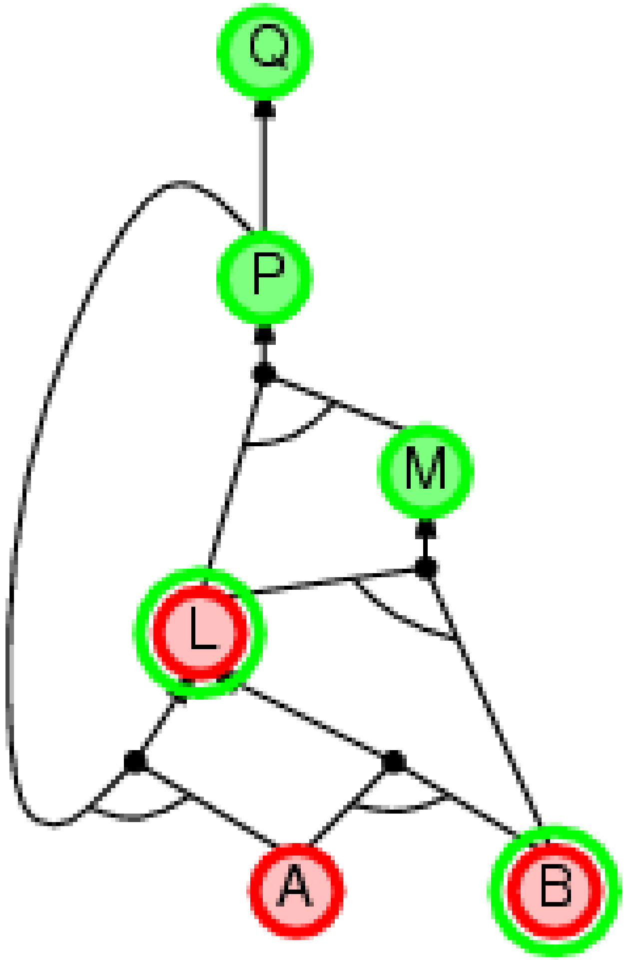


As soon as you can move forward, do so.

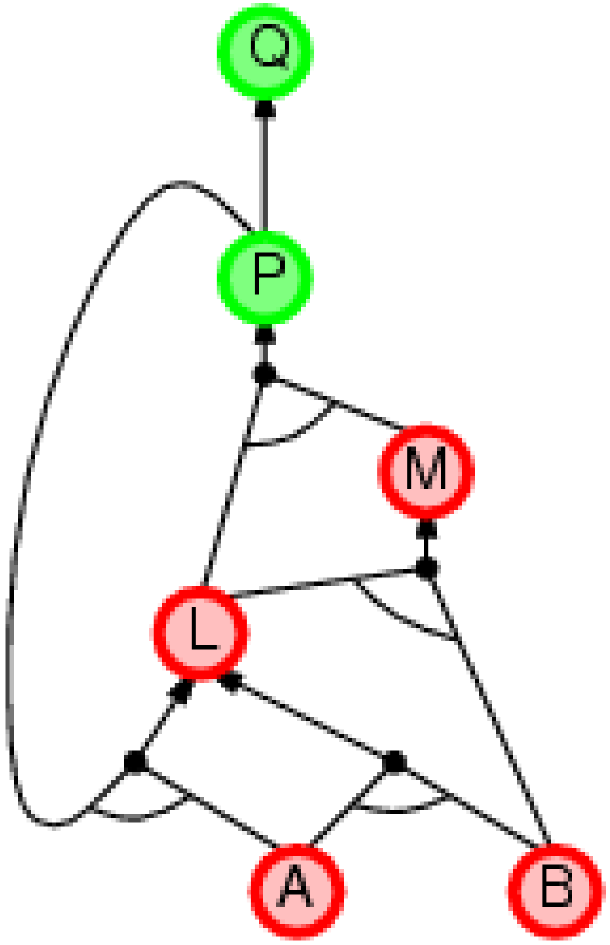
Backward chaining example



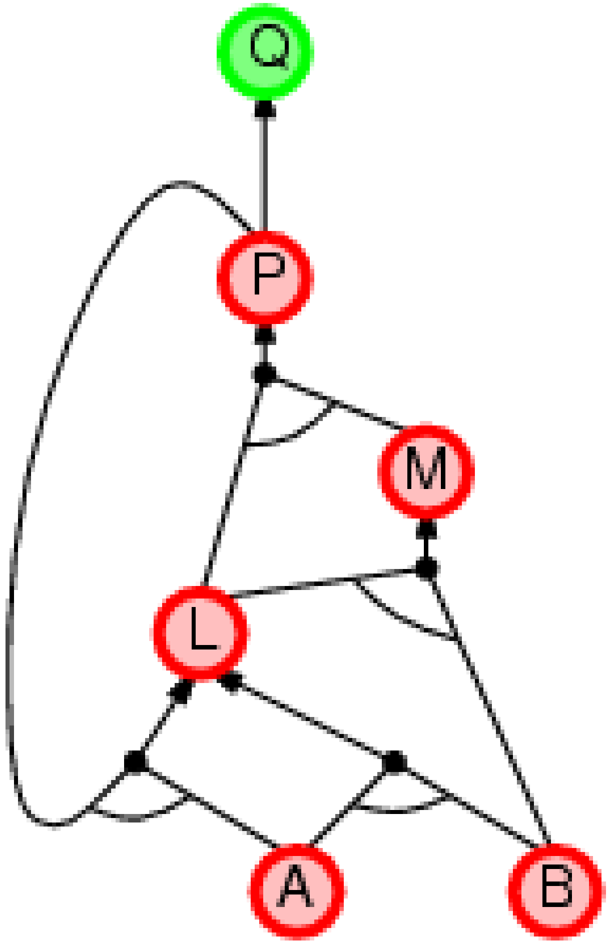
Backward chaining example



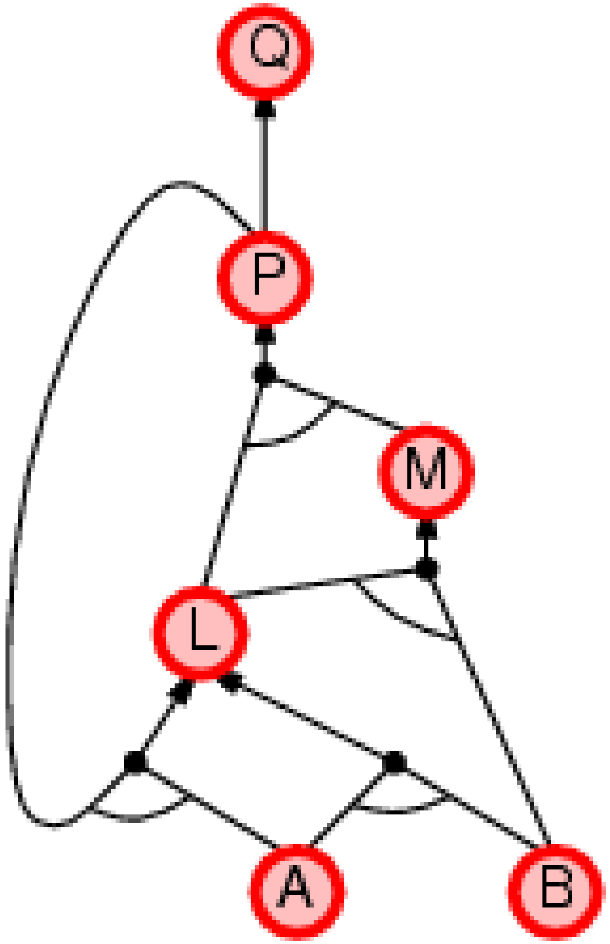
Backward chaining example



Backward chaining example



Backward chaining example



Forward vs. backward chaining

- FC is data-driven, automatic, unconscious processing,
  - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be much less than linear in size of KB

Model Checking

Two families of efficient algorithms:

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

The DPLL algorithm

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

Improvements:

1. Early termination

   A clause is true if any literal is true.

   A sentence is false if any clause is false.

2. Pure symbol heuristic

   Pure symbol: always appears with the same "sign" in all clauses.

   e.g., In the three clauses $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A and B are pure, C is impure.

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

3 Unit clause heuristic

   Unit clause: only one literal in the clause

   The only literal in a unit clause must be true.

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

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

$$A = pure$$

The `WalkSAT` algorithm

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

Walksat Procedure

Start with random initial assignment.

Pick a random unsatisfied clause.

Select and flip a variable from that clause:

- With probability p, pick a random variable.
- With probability 1-p, pick greedily
  - a variable that minimizes the number of unsatisfied clauses

Repeat to predefined maximum number flips;
if no solution found, restart.

Hard satisfiability problems

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

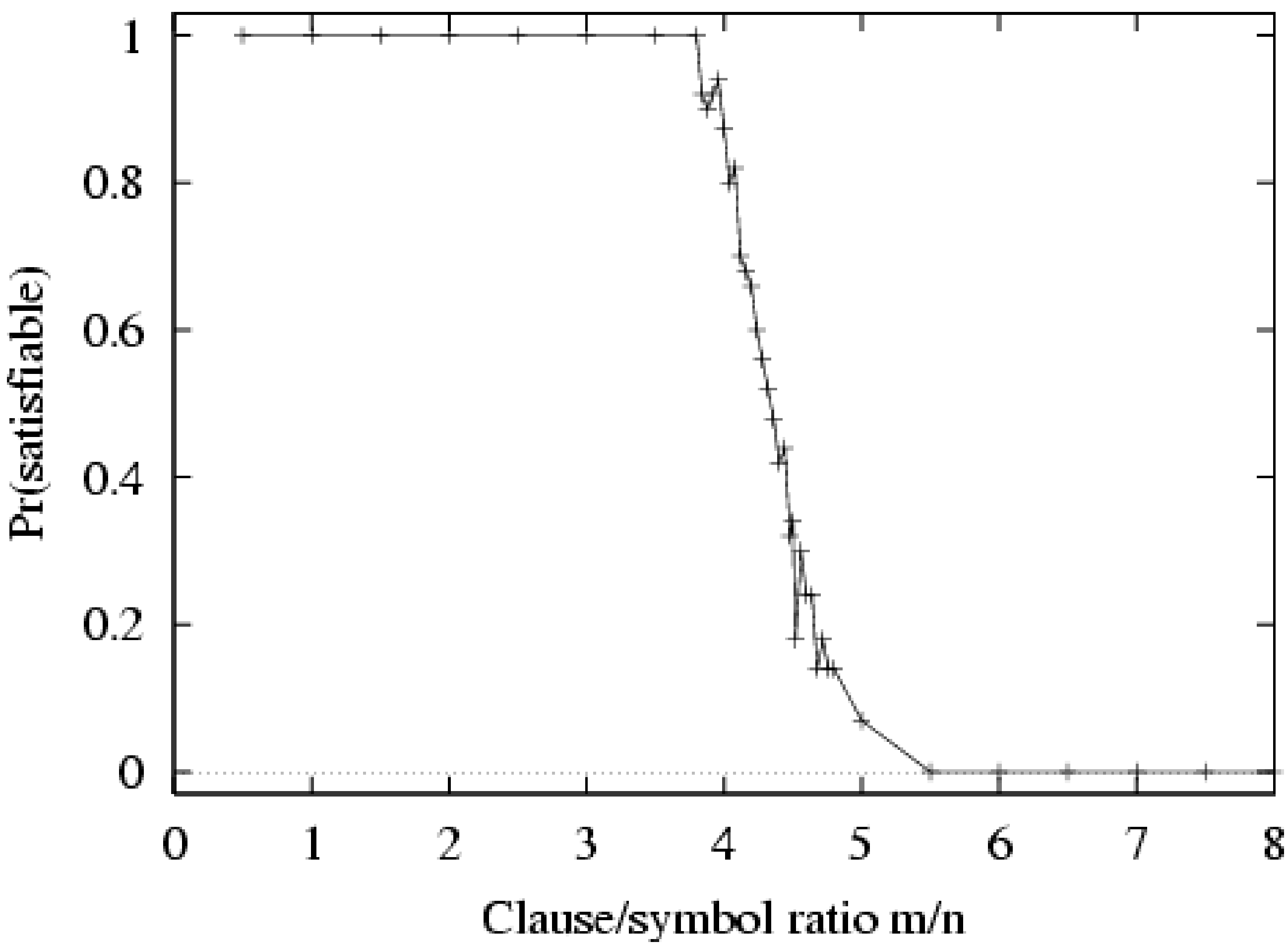
  $(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)$

  m = number of clauses (5)

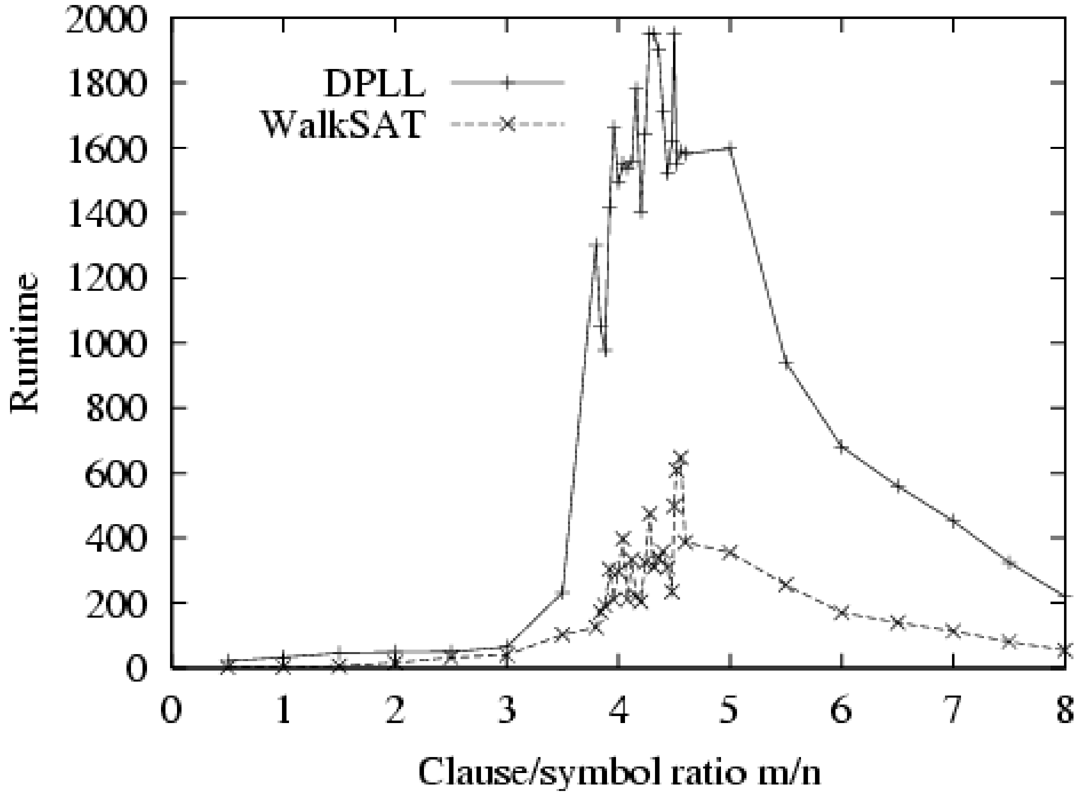
  n = number of symbols (5)

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

Hard satisfiability problems



Hard satisfiability problems



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

Hardness of CSPs

- $x_1 \ldots x_n$ discrete, domain size d: O(d^n) configurations
- "SAT": Boolean satisfiability: d=2
  - The first known NP-complete problem
- "3-SAT"
  - Conjunctive normal form (CNF)
  - At most 3 variables in each clause:

    $$(x_1 \vee \neg x_7 \vee x_{12}) \wedge (\neg x_3 \vee x_2 \vee x_7) \wedge \ldots$$

  - Still NP-complete

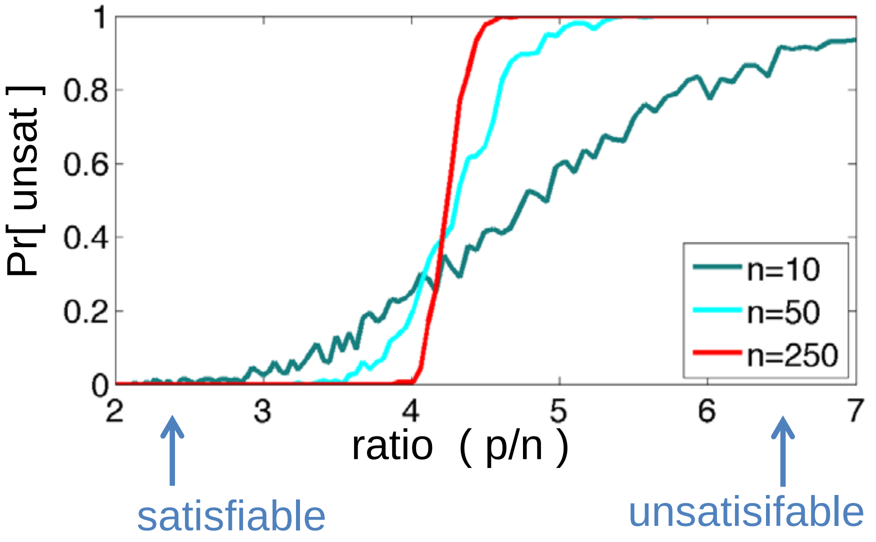
CNF clause: rule out one configuration

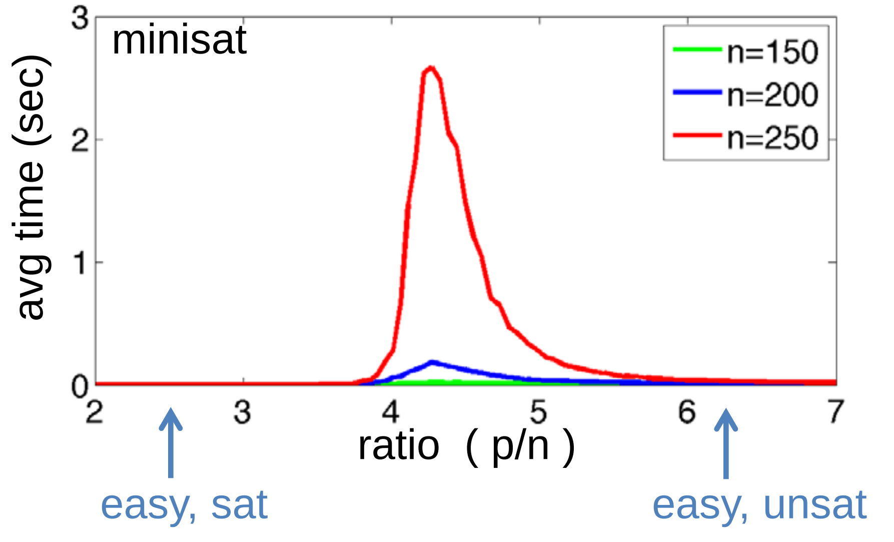
- How hard are "typical" problems?

Hardness of random CSPs

- Random 3-SAT problems:
  - n variables, p clauses in CNF: $(x_1 \vee \neg x_7 \vee x_{12}) \wedge (\neg x_3 \vee x_2 \vee x_7) \wedge \ldots$
  - Choose any 3 variables, signs uniformly at random
  - What's the probability there is **no** solution to the CSP?

  - Phase transition at (p/n) ¼ 4.25
  - "Hard" instances fall in a very narrow regime around this point!

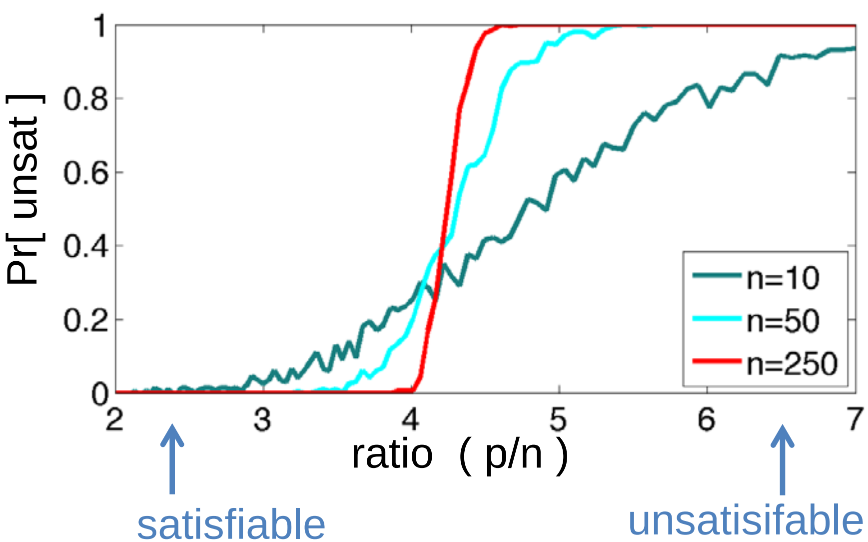


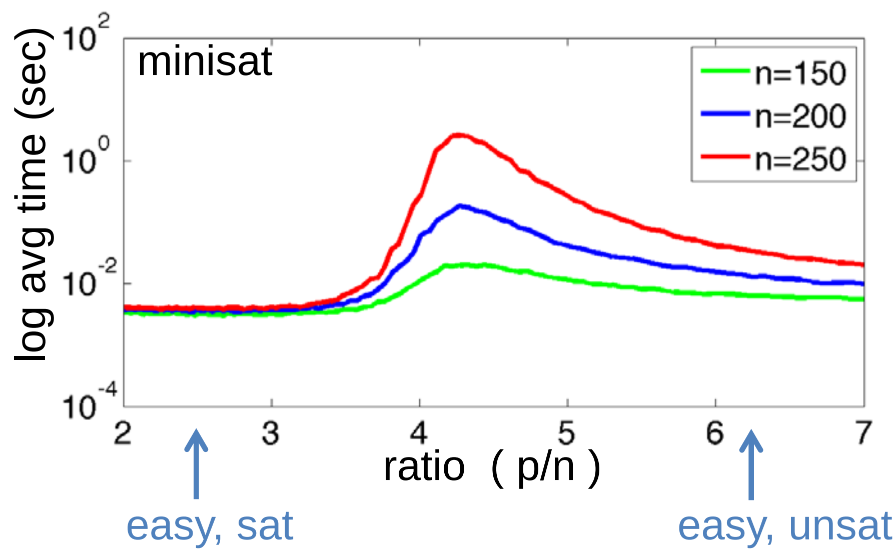


Hardness of random CSPs

- Random 3-SAT problems:
  - n variables, p clauses in CNF: $(x_1 \vee \neg x_7 \vee x_{12}) \wedge (\neg x_3 \vee x_2 \vee x_7) \wedge \ldots$
  - Choose any 3 variables, signs uniformly at random
  - What's the probability there is **no** solution to the CSP?

  - Phase transition at (p/n) ¼ 4.25
  - "Hard" instances fall in a very narrow regime around this point!





Common Sense Reasoning

Example, adapted from Lenat

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

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

- Can Propositional Logic support these inferences?

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions

- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences

- Resolution is complete for propositional logic.
  Forward and backward chaining are linear-time, complete for Horn clauses

- Propositional logic lacks expressive power