Logic Agents and Propositional Logic

CHAPTER 7

HASSAN KHOSRAVI

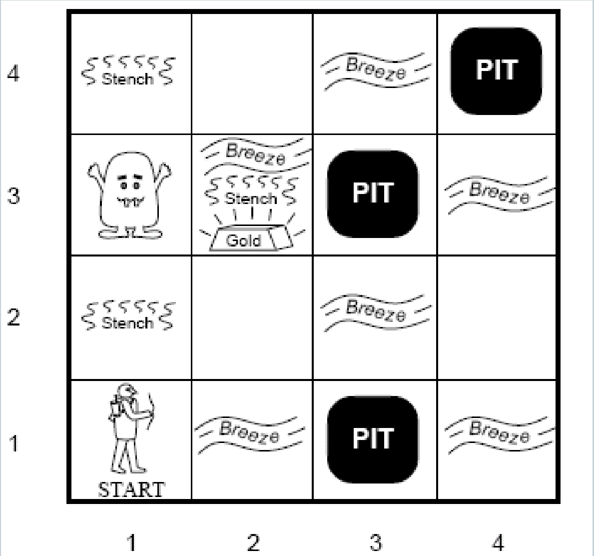
SPRING2011

Knowledge-Based Agents

- KB = knowledge base
  - A set of sentences or facts
  - e.g., a set of statements in a logic language
- Inference
  - Deriving new sentences from old
  - e.g., using a set of logical statements to infer new ones
- A simple model for reasoning
  - Agent is told or perceives new evidence
    - E.g., A is true
  - Agent then infers new facts to add to the KB
    - E.g., KB = { A -> (B OR C) }, then given A and not C we can infer that B is true
    - B is now added to the KB even though it was not explicitly asserted, i.e., the agent inferred B

Wumpus World

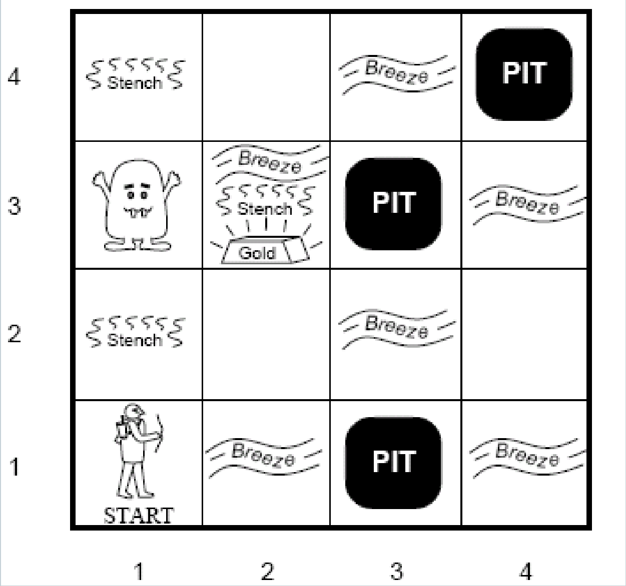
- Environment
  - Cave of 4×4
  - Agent enters in [1,1]
  - 16 rooms
    - Wumpus: A deadly beast who kills anyone entering his room.
    - Pits: Bottomless pits that will trap you forever.
    - Gold



Wumpus World

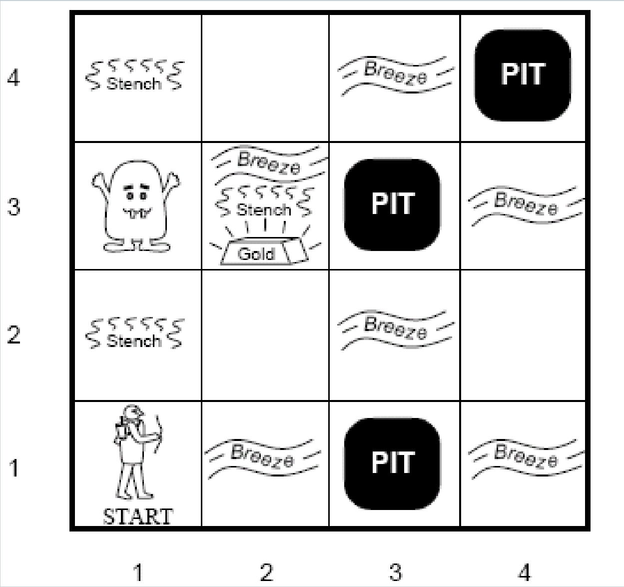
- Agents Sensors:
  - Stench next to Wumpus
  - Breeze next to pit
  - Glitter in square with gold
  - Bump when agent moves into a wall
  - Scream from wumpus when killed

- Agents actions
  - Agent can move forward, turn left or turn right
  - Shoot, one shot



Wumpus World

- Performance measure
  - +1000 for picking up gold
  - -1000 got falling into pit
  - -1 for each move
  - -10 for using arrow



Reasoning in the Wumpus World

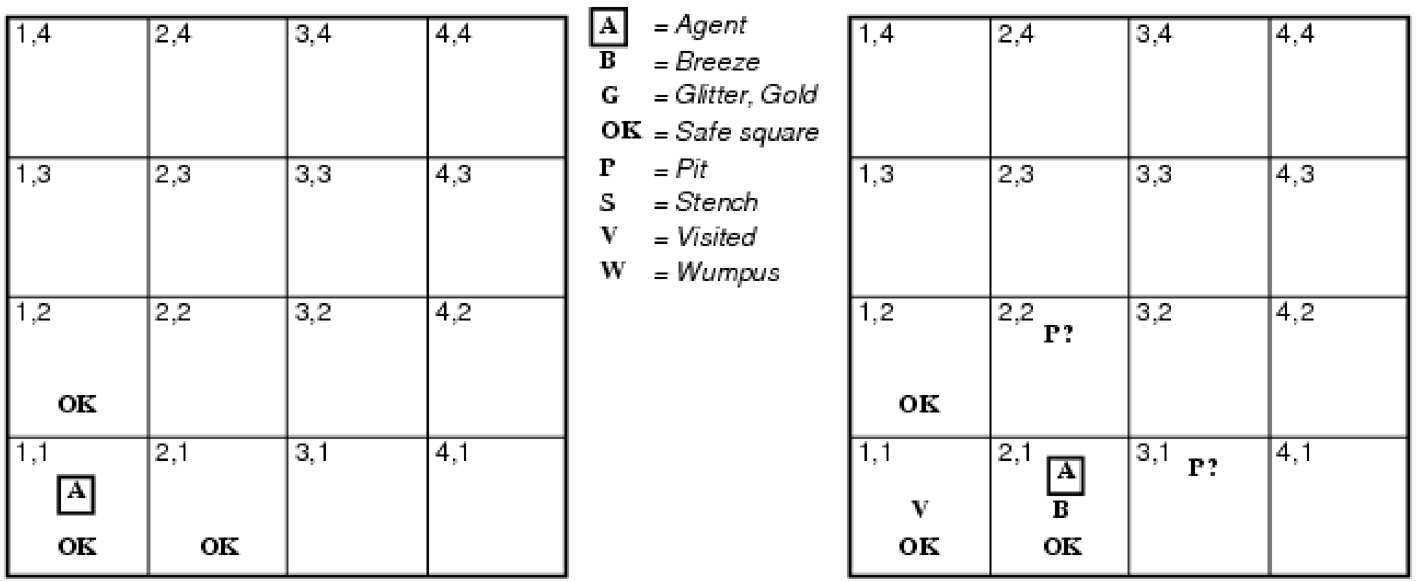
- Agent has initial ignorance about the configuration
  - Agent knows his/her initial location
  - Agent knows the rules of the environment

- Goal is to explore environment, make inferences (reasoning) to try to find the gold.

- Random instantiations of this problem used to test agent reasoning and decision algorithms

(applications? "intelligent agents" in computer games)

Exploring the Wumpus World



(a) (b)

[1,1] The KB initially contains the rules of the environment.

The first percept is [*none, none,none,none,none*],

move to safe cell e.g. 2,1

Exploring the Wumpus World

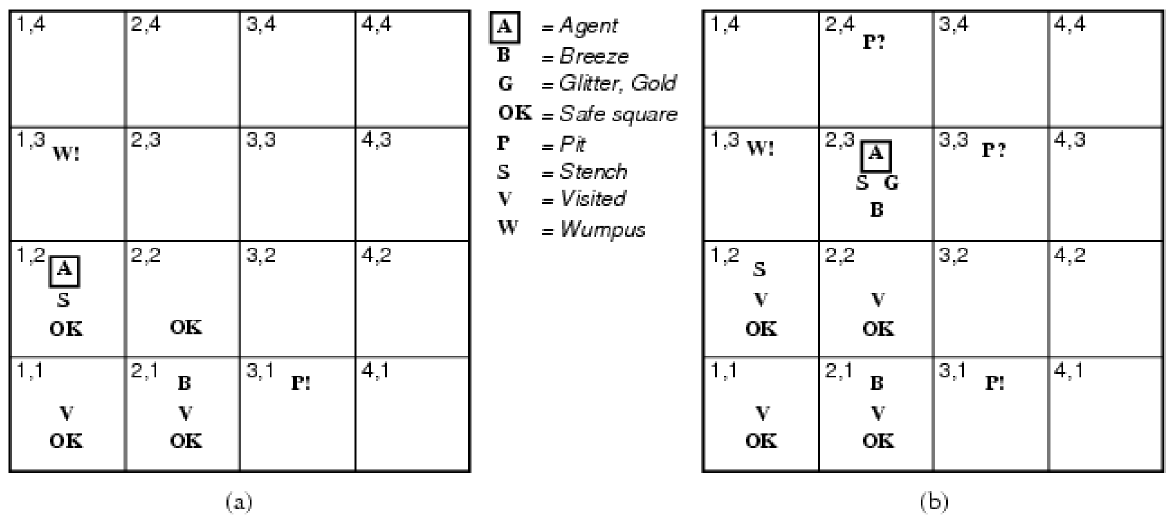
(a)
(b)

[2,1] = breeze

indicates that there is a pit in [2,2] or [3,1],

return to [1,1] to try next safe cell

Exploring the Wumpus World



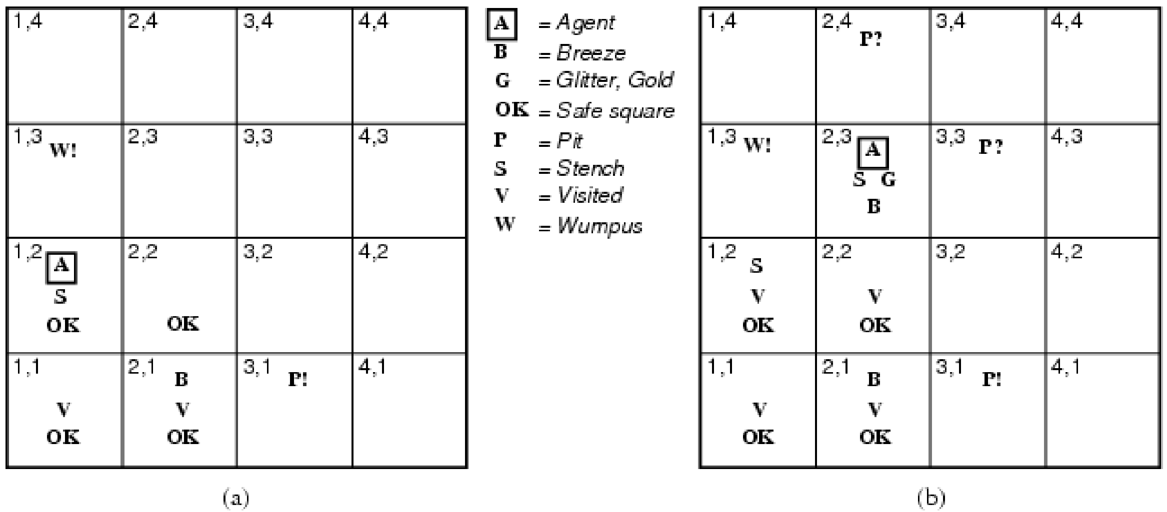
[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]

YET … not in [1,1]

YET … not in [2,2] or stench would have been detected in [2,1]

(this is relatively sophisticated reasoning!)

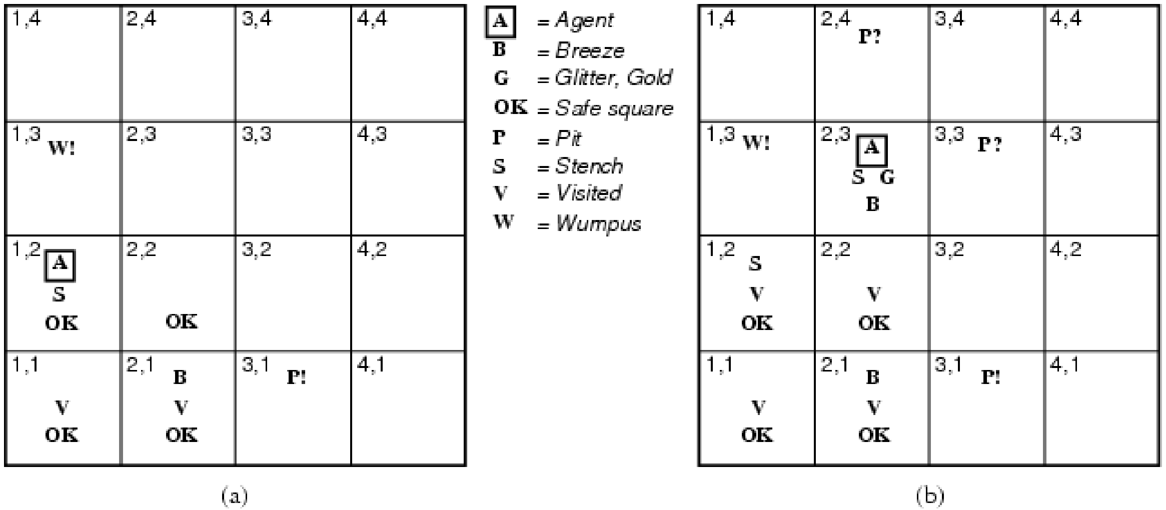
Exploring the Wumpus World



[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]
YET … not in [1,1]
YET … not in [2,2] or stench would have been detected in [2,1]
(this is relatively sophisticated reasoning!)

THUS … wumpus is in [1,3]
THUS [2,2] is safe because of lack of breeze in [1,2]
THUS pit in [1,3] (again a clever inference)
move to next safe cell [2,2]

Exploring the Wumpus World



[2,2] move to [2,3]

[2,3] detect glitter , smell, breeze
THUS pick up gold
THUS pit in [3,3] or [2,4]

What our example has shown us

- Can represent general knowledge about an environment by a set of rules and facts
- Can gather evidence and then infer new facts by combining evidence with the rules
- The conclusions are guaranteed to be correct if
  - The evidence is correct
  - The rules are correct
  - The inference procedure is correct

    -> logical reasoning
- The inference may be quite complex
  - E.g., evidence at different times, combined with different rules, etc

What is a Logic?

- A formal language
  - KB = set of sentences
- Syntax
  - what sentences are legal (well-formed)
  - E.g., arithmetic
    - X+2 >= y is a wf sentence, +x2y is not a wf sentence
- Semantics
  - loose meaning: the interpretation of each sentence
  - More precisely:
    - Defines the truth of each sentence wrt to each possible world
  - e.g,
    - X+2 = y is true in a world where x=7 and y =9
    - X+2 = y is false in a world where x=7 and y =1
  - Note: standard logic – each sentence is T of F wrt eachworld
    - Fuzzy logic – allows for degrees of truth.

Models and possible worlds

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.
- m is a model of a sentence α if α is true in m
- $M(\alpha)$ is the set of all models of α
- Possible worlds ~ models
  - Possible worlds: potentially real environments
  - Models: mathematical abstractions that establish the truth or falsity of every sentence
- Example:
  - x + y = 4, where x = #men, y = #women
  - Possible models = all possible assignments of integers to x and y

Entailment

- One sentence follows logically from another

  $$\alpha \models \beta$$

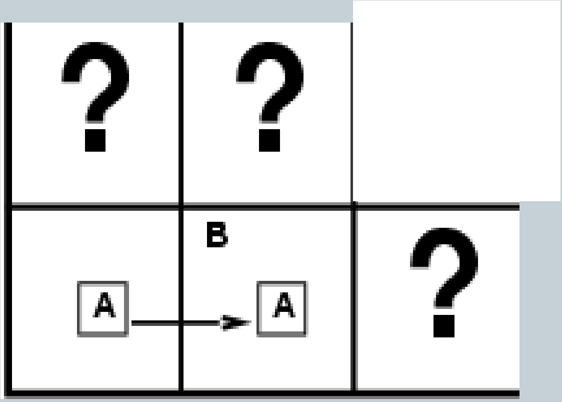
  α entails sentence β *if and only if* β is true in all worlds where α is true.

  e.g., $x+y=4 \models 4=x+y$

- Entailment is a relationship between sentences that is based on semantics.

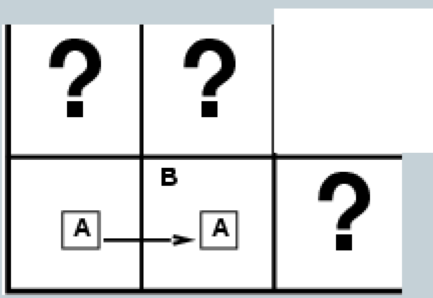
Entailment in the wumpus world

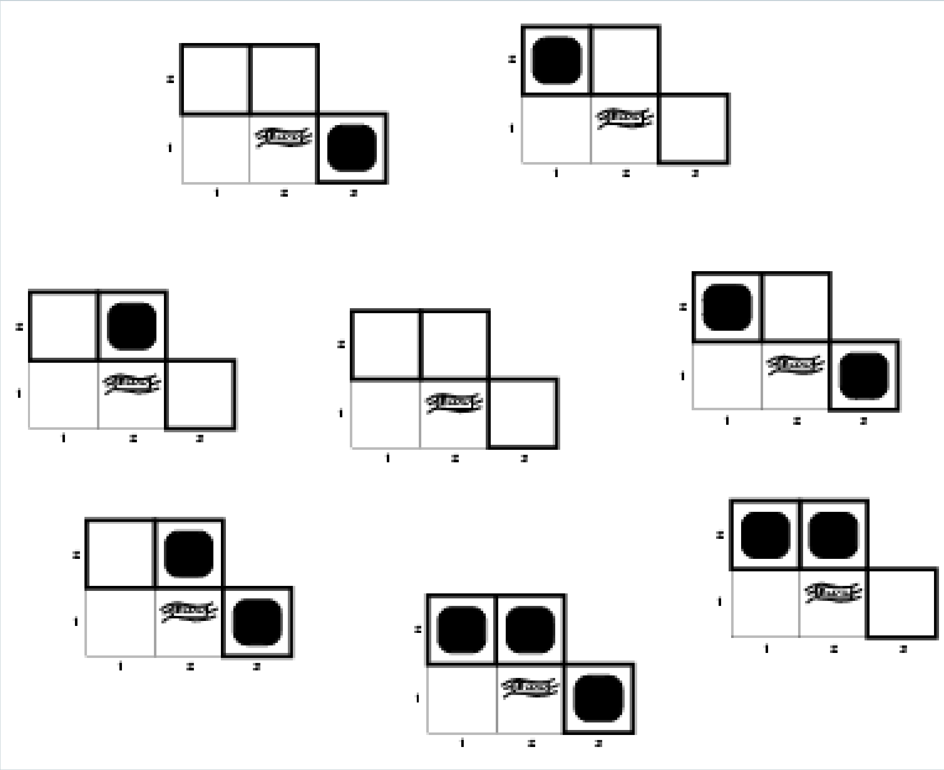
- Consider possible models for *KB* assuming only pits and a reduced Wumpus world
- Situation after detecting nothing in [1,1], moving right, detecting breeze in [2,1]



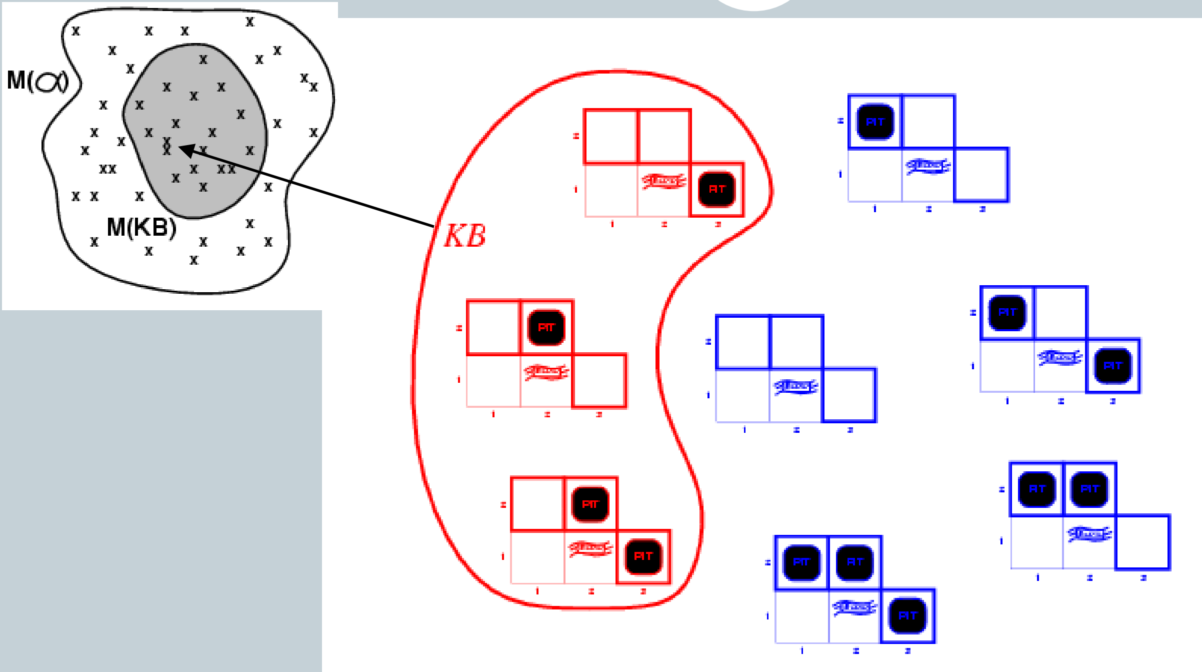
Wumpus models

All possible models in this reduced Wumpus world.





Wumpus models

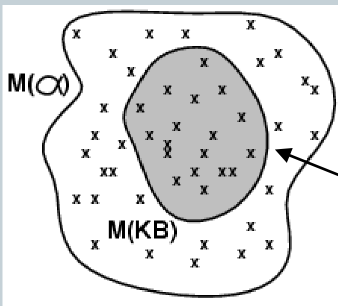


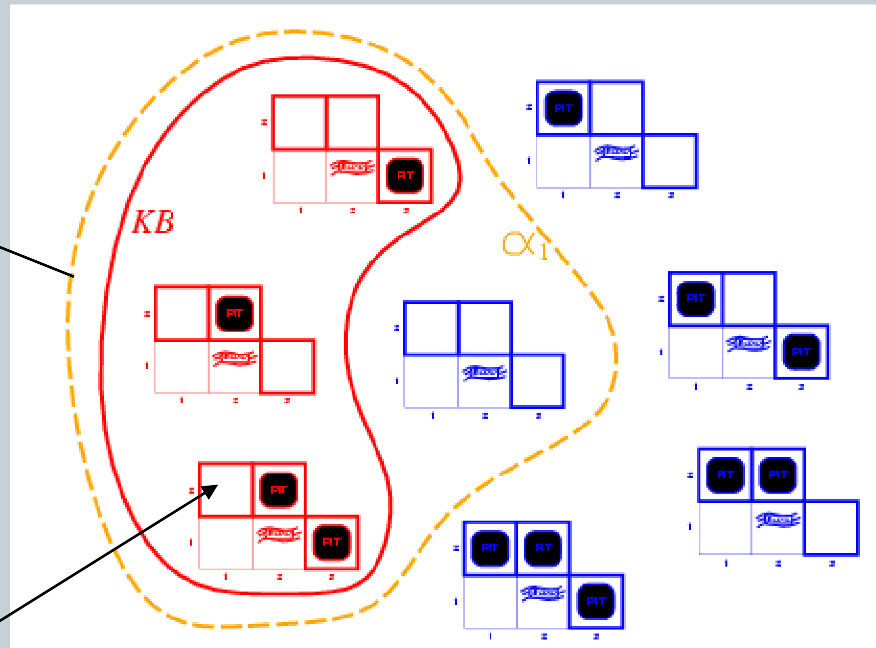
- *KB* = all possible wumpus-worlds consistent with the observations and the "physics" of the Wumpus world.

Inferring conclusions

- Consider 2 possible conclusions given a KB
  - α_1 = "[1,2] is safe"
  - α_2 = "[2,2] is safe"
- One possible inference procedure
  - Start with KB
  - Model-checking
    - Check if KB $\models \alpha$ by checking if in all possible models where KB is true that α is also true
- Comments:
  - Model-checking enumerates all possible worlds
    - Only works on finite domains, will suffer from exponential growth of possible models

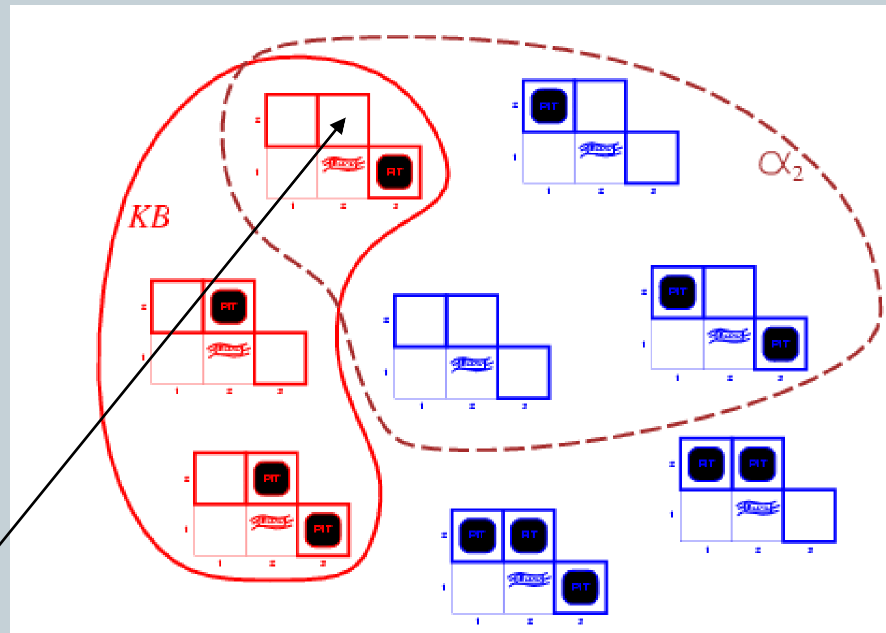
Wumpus models





α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

There are some models entailed by KB where α_2 is false

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.

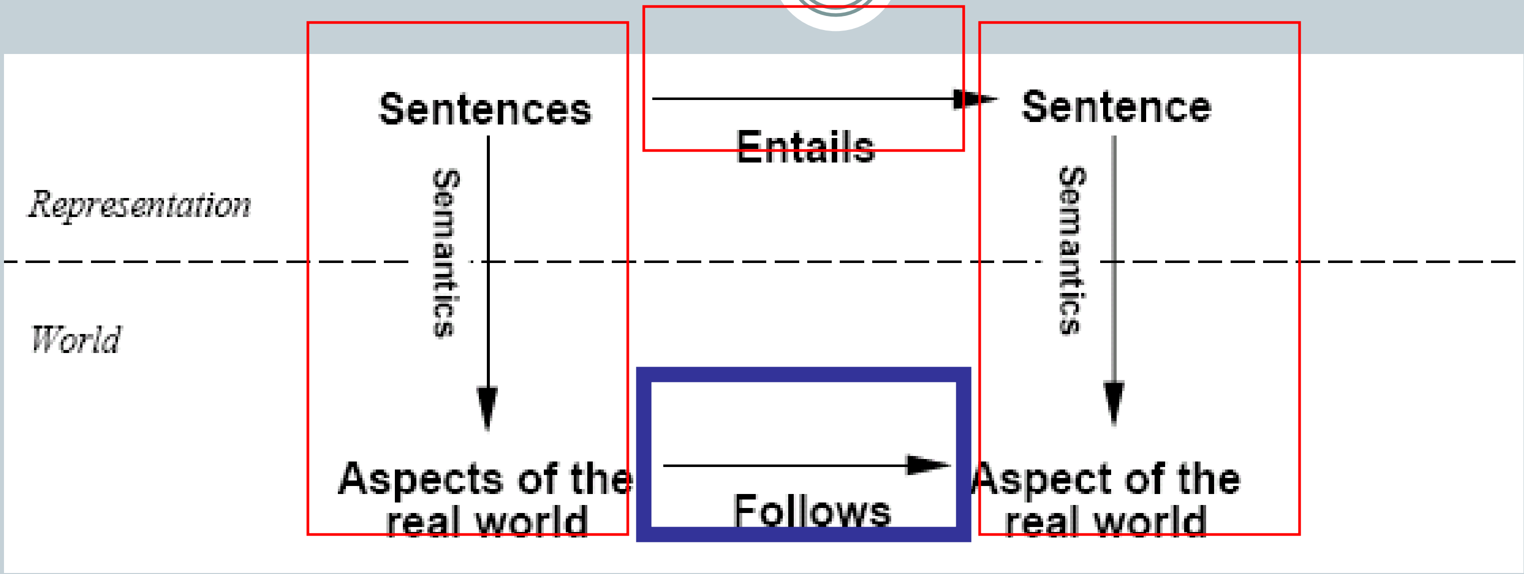
- If an algorithm only derives entailed sentences it is called *sound* or *truth preserving*.
  - Otherwise it just makes things up.

    i is sound if whenever KB $|-_i \alpha$ *it is also true that KB* $|= \alpha$
  - *E.g., model-checking is sound*

- Completeness : the algorithm can derive any sentence that is entailed.

  i is complete if whenever KB $|= \alpha$ *it is also true that KB* $|-_i \alpha$

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.

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- Atomic sentences = single proposition symbols
  - E.g., P, Q, R
  - Special cases: True = always true, False = always false
- Complex sentences:
  - If S is a sentence, $\neg S$ is a sentence (negation)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic: Semantics

Each model/world specifies true or false for each proposition symbol

E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$

false true false

With these symbols, 8 possible models, can be enumerated automatically.

Rules for evaluating truth with respect to a model *m*:

$\neg S$	is true iff	S is false	
$S_1 \wedge S_2$	is true iff	S_1 is true and	S_2 is true
$S_1 \vee S_2$	is true iff	S_1 is true or	S_2 is true
$S_1 \Rightarrow S_2$	is true iff	S_1 is false or	S_2 is true
i.e.,	is false iff	S_1 is true and	S_2 is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true and	$S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \mathit{true} \wedge (\mathit{true} \vee \mathit{false}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Implication is always true when the premise is false

Why? P=>Q means "if P is true then I am claiming that Q is true otherwise no claim"

Only way for this to be false is if P is true and Q is false

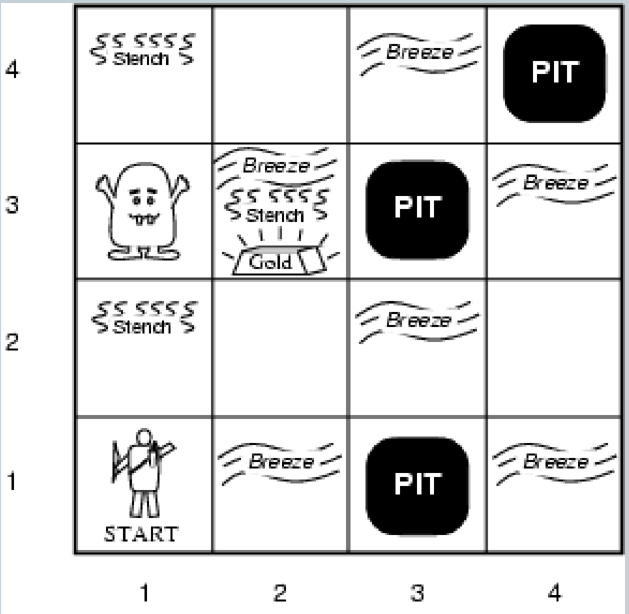
Wumpus world sentences

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

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

$\neg B_{1,1}$

$B_{2,1}$



- "Pits cause breezes in adjacent squares"

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

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

- KB can be expressed as the conjunction of all of these sentences

- Note that these sentences are rather long-winded!
  - E.g., breese "rule" must be stated explicitly for each square
  - First-order logic will allow us to define more general relations (later)

Truth tables for the Wumpus KB

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

Inference by enumeration

- We want to see if α is entailed by KB
- Enumeration of all models is sound and complete.
- But…for n symbols, time complexity is $O(2^n)$…
- We need a more efficient way to do inference
  - But worst-case complexity will remain exponential for propositional logic

Logical equivalence

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

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

- Modus Ponens

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

- And-Elimination

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

- Bi-conditional Elimination

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

Validity and satisfiability

A sentence is valid if it is true in all models,
e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
(tautologies)

Validity is connected to inference via the Deduction Theorem:
$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model
e.g., $A \vee B$, C
(determining satisfiability of sentences is NP-complete)

A sentence is unsatisfiable if it is false in all models
e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:
$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable
(there is no model for which KB=true and α is false)
(aka proof by contradiction: assume α to be false and this leads to contraditions in KB)

Proof methods

- Proof methods divide into (roughly) two kinds:

Application of inference rules:

Legitimate (sound) generation of new sentences from old.

- Resolution
- Forward & Backward chaining

Model checking

Searching through truth assignments.

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

Normal Form

We want to prove:

$$KB \models \alpha$$

$$\textit{equivalent to}: KB \wedge \neg\alpha \textit{ unsatifiable}$$

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

A "conjunction of disjunctions"

literals

$$\underbrace{(A \vee \neg B)}_{\text{Clause}} \wedge \underbrace{(B \vee \neg C \vee \neg D)}_{\text{Clause}}$$

- Any KB can be converted into CNF
- k-CNF: exactly k literals per clause

Example: Conversion to CNF

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

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

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

3. Move $\neg$ inwards using de Morgan's rules and double-negation:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

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

Resolution Inference Rule for CNF

$(A \vee B \vee C)$

$(\neg A)$

$- - - - - - - - - - - - - -$

$\therefore (B \vee C)$

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

$(A \vee B \vee C)$

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

$- - - - - - - - - - - - -$

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

"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

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 query.
- One of two things can happen:

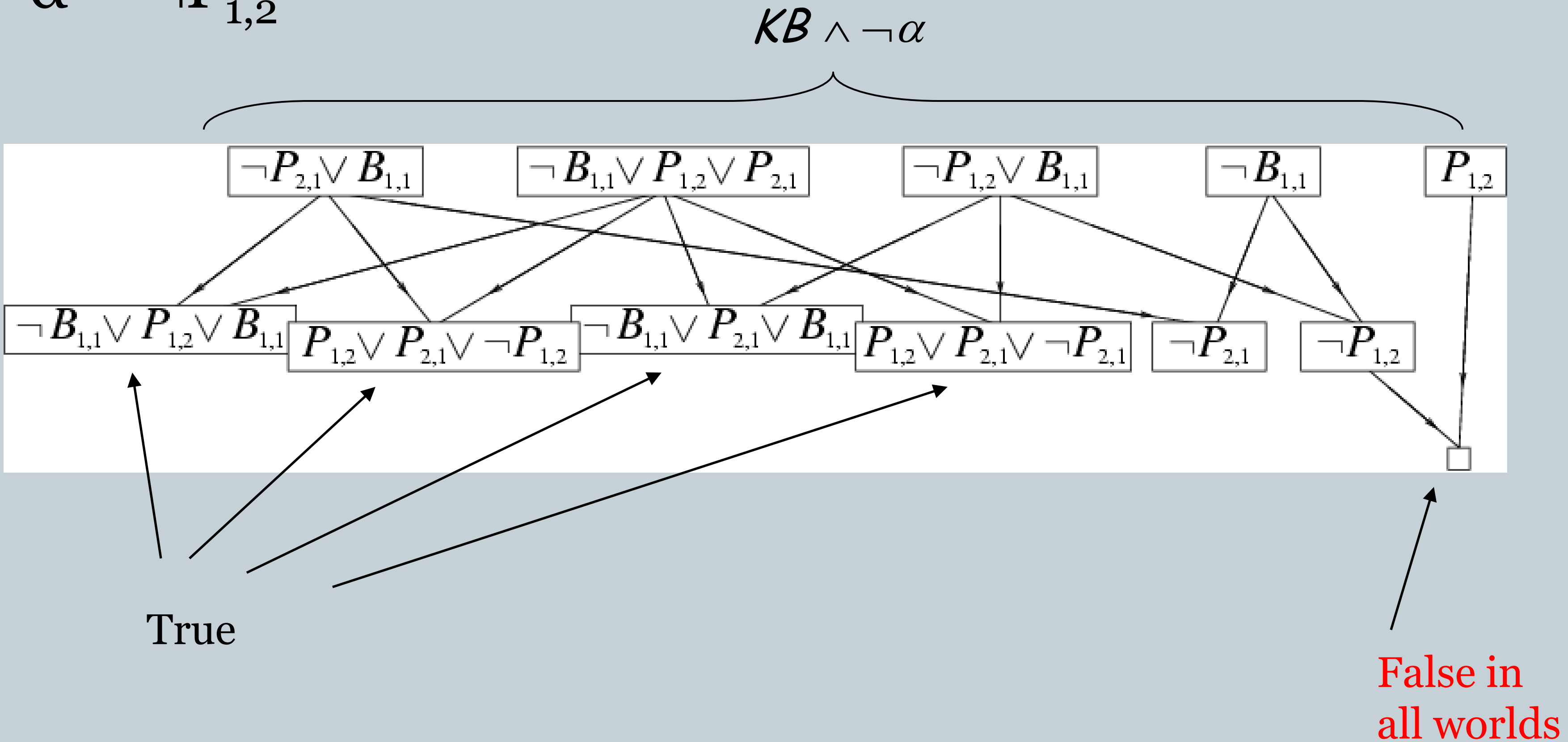
1. We find $P \wedge \neg P$ which is unsatisfiable,
   i.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the
   Sentence (non-trivial) and hence we cannot entail the query.

$KB \wedge \neg\alpha$

Resolution example

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



Horn Clauses

- Resolution in general can be exponential in space and time.

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

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

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

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

- 1 positive literal: definite clause

- 0 positive literals: Fact or integrity constraint:
  e.g. $(\neg A \vee \neg B) \equiv (A \wedge B \Rightarrow False)$

Forward-chaining pseudocode

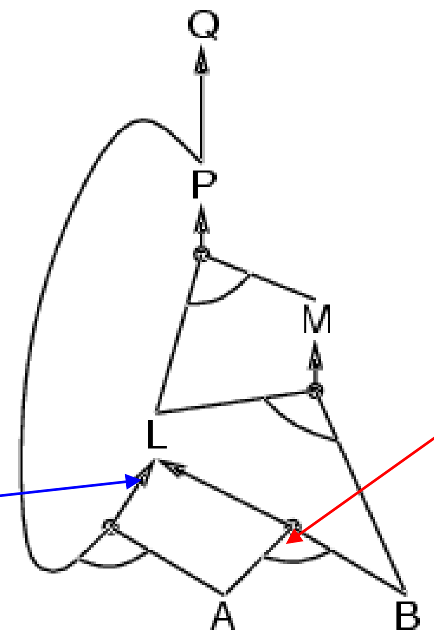
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

Forward chaining: graph representation

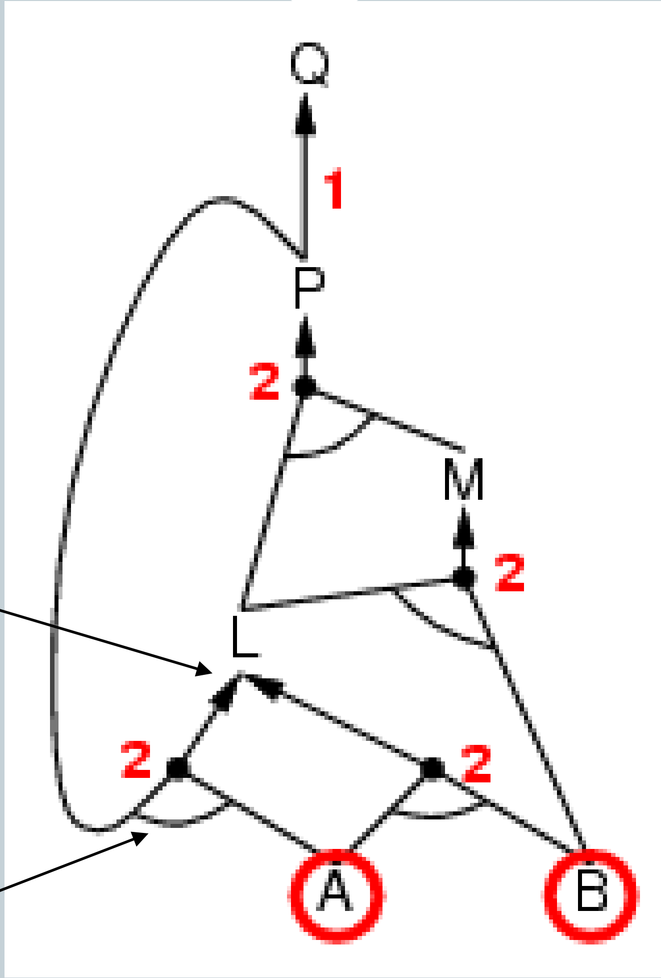
- Idea: fire any rule whose premises are satisfied in the *KB*,
  - add its conclusion to the *KB*, until query is found

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



- Forward chaining is sound and complete for Horn KB

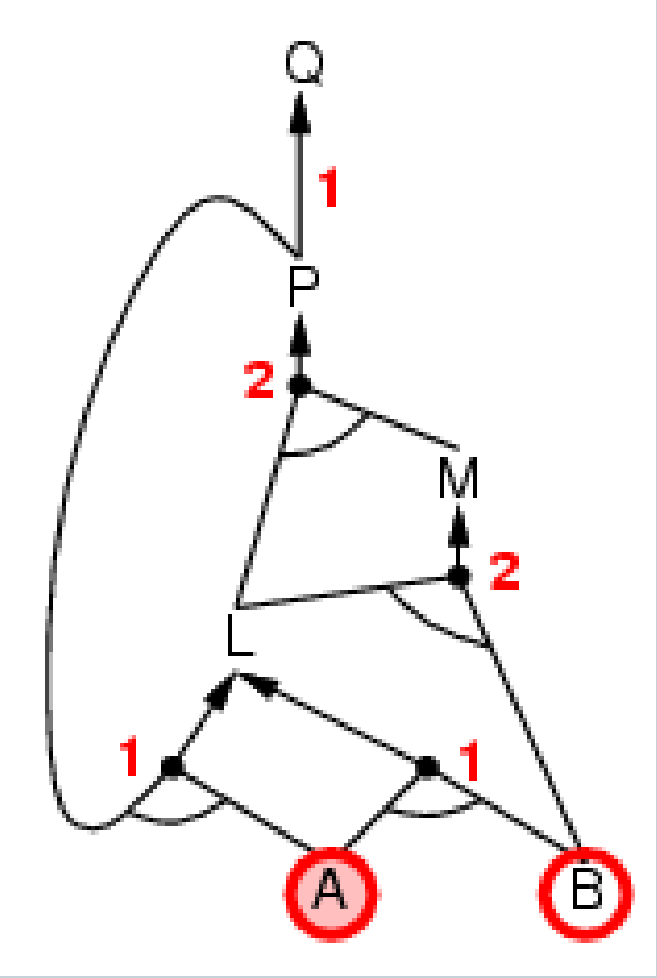
Forward chaining example



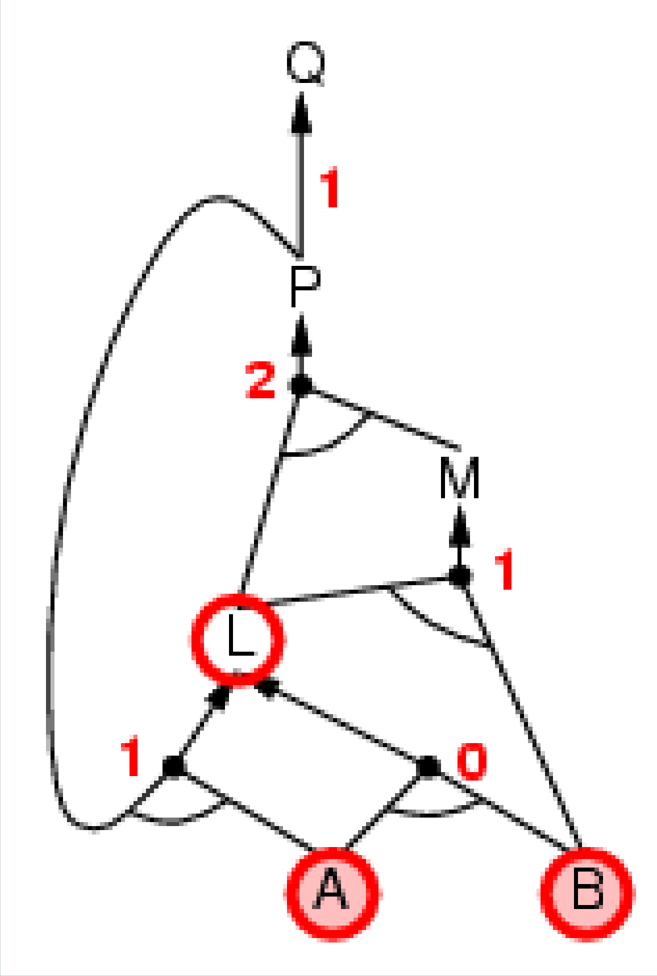
"OR" Gate

"AND" gate

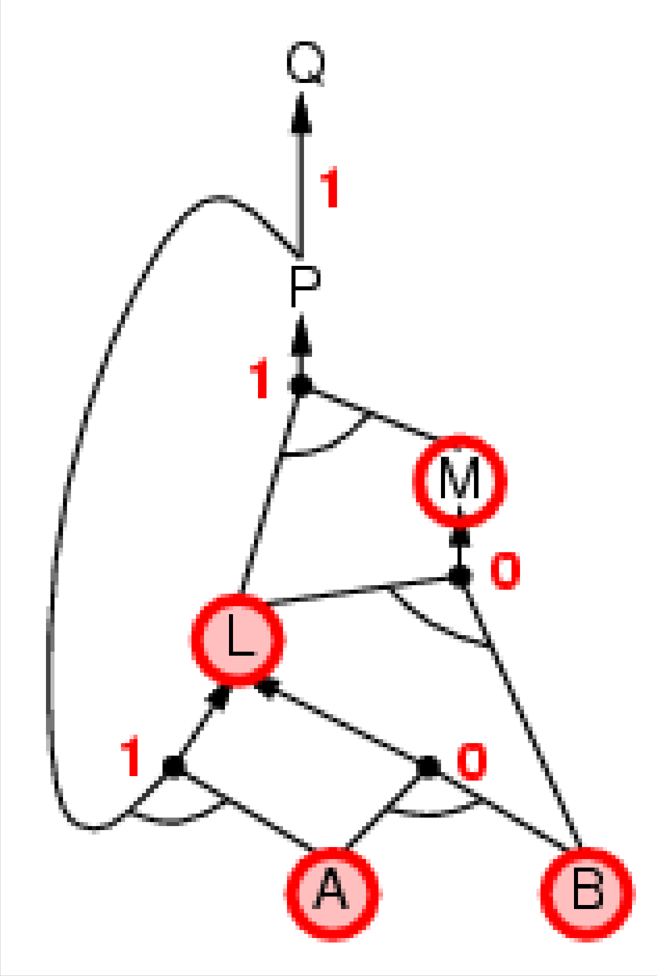
Forward chaining example



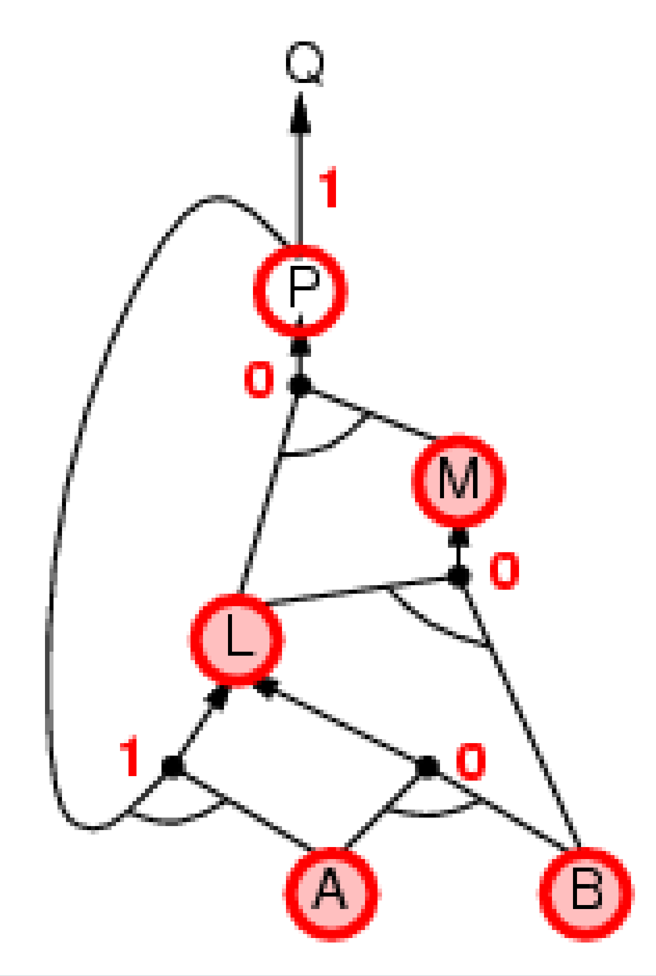
Forward chaining example



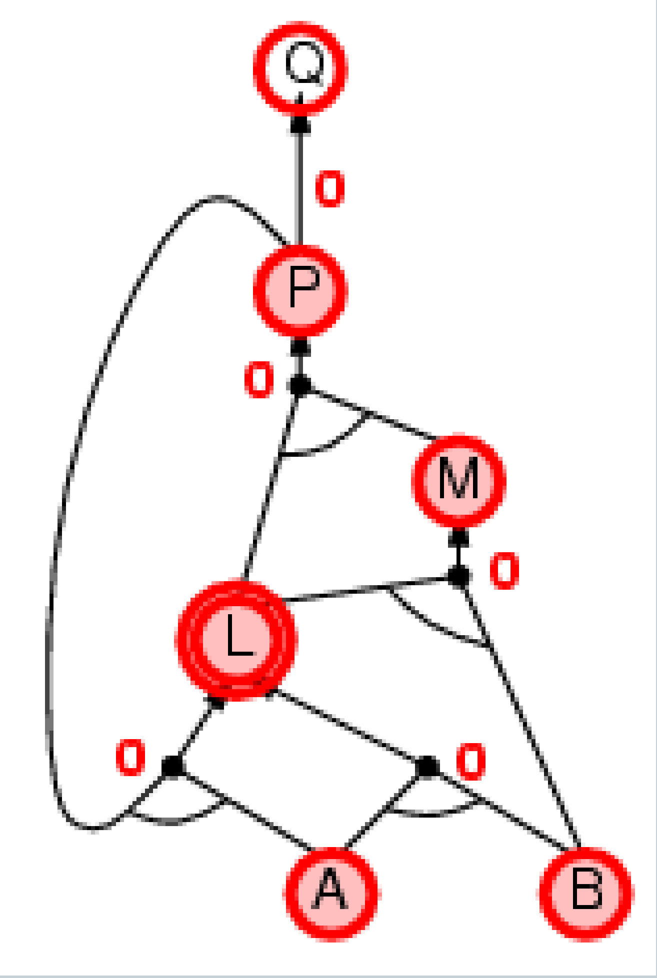
Forward chaining example



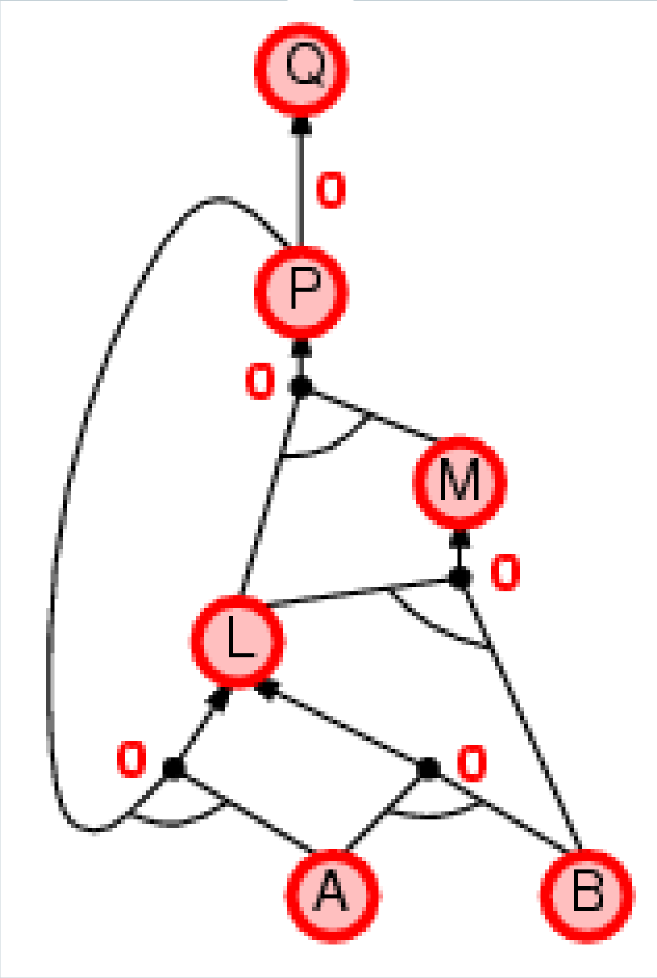
Forward chaining example



Forward chaining example



Forward chaining example



Forward chaining

- FC is data-driven, automatic, unconscious processing,
  - e.g., object recognition, routine decisions

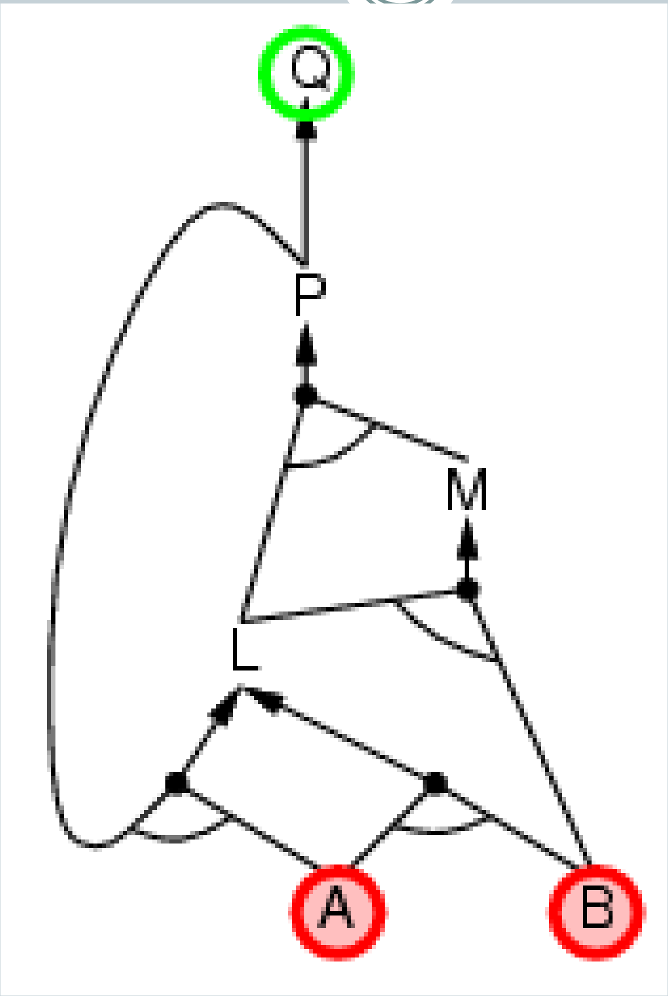
- May do lots of work that is irrelevant to the goal

Backward chaining

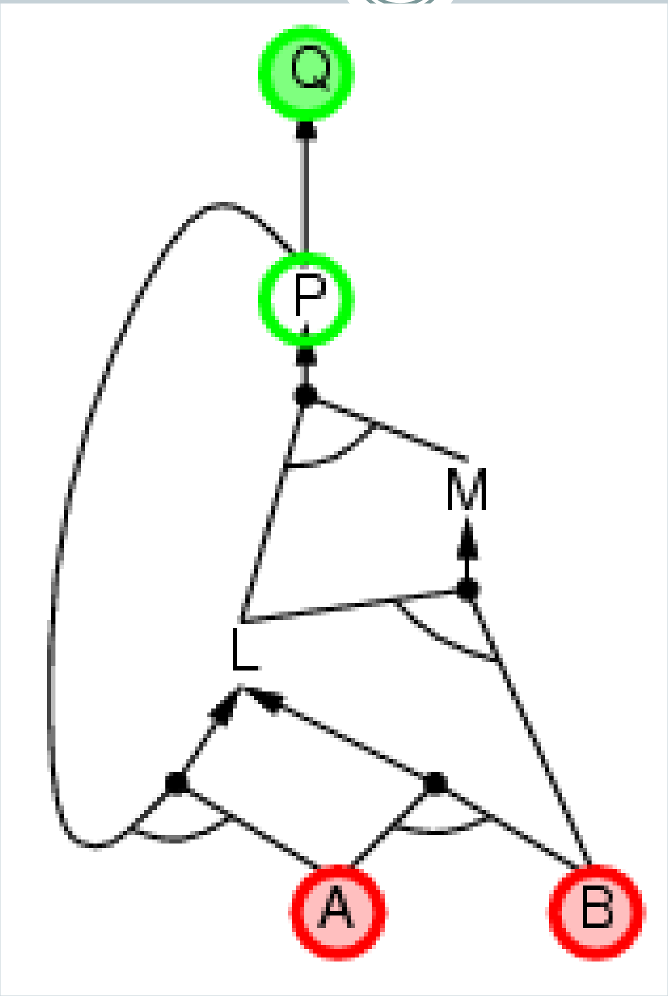
Idea: work backwards from the query *q*

- 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.

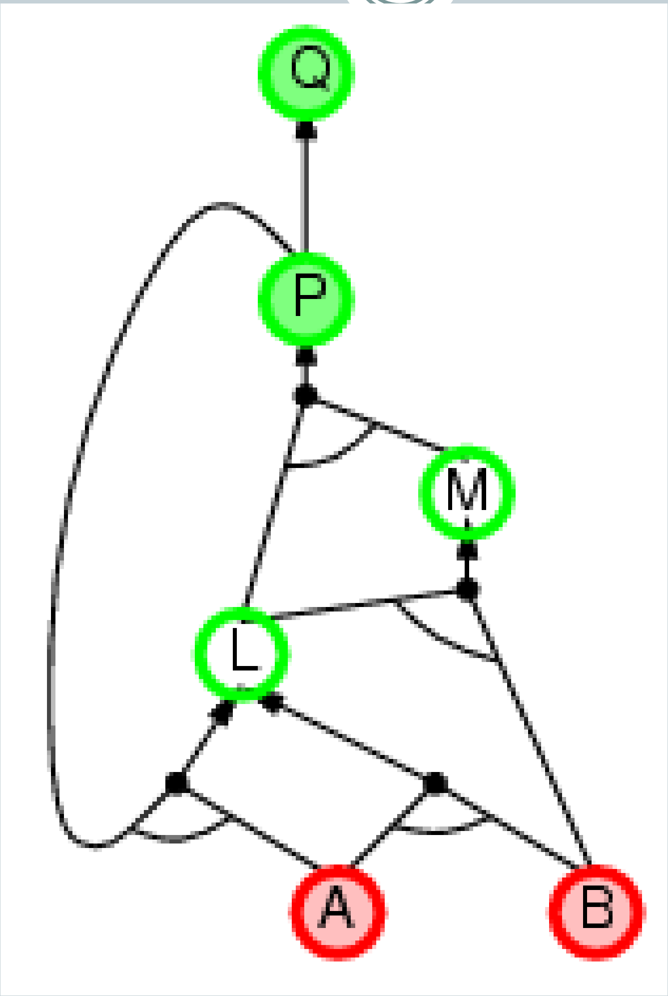
Backward chaining example



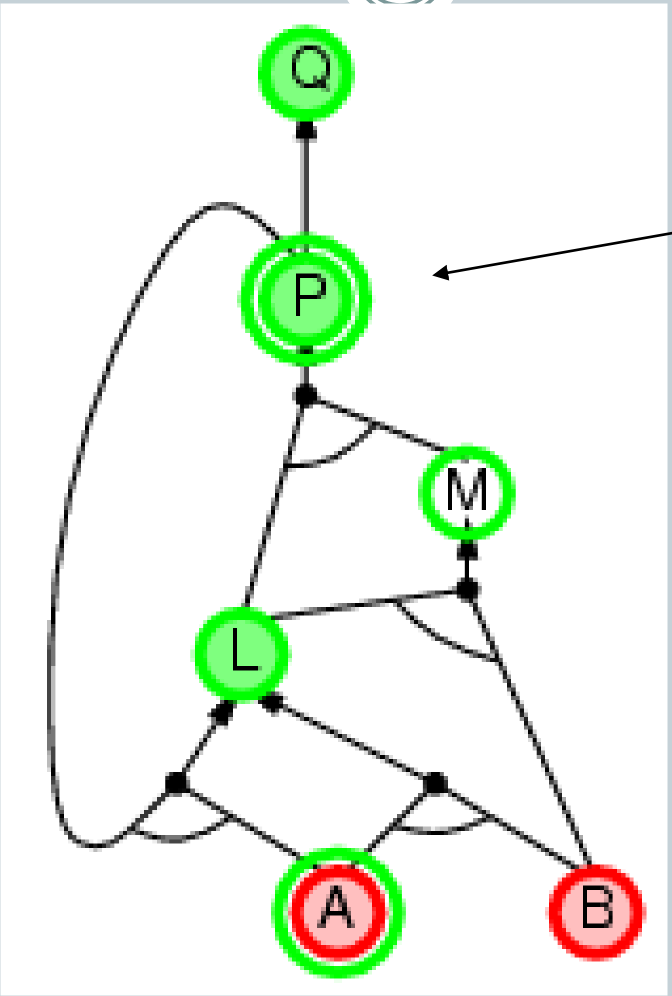
Backward chaining example



Backward chaining example

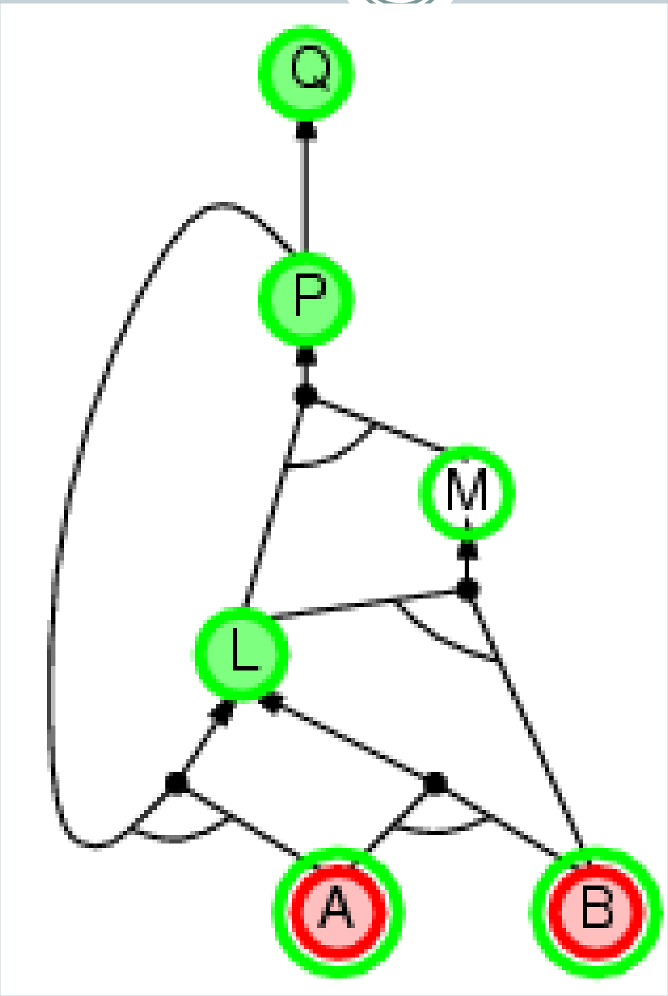


Backward chaining example

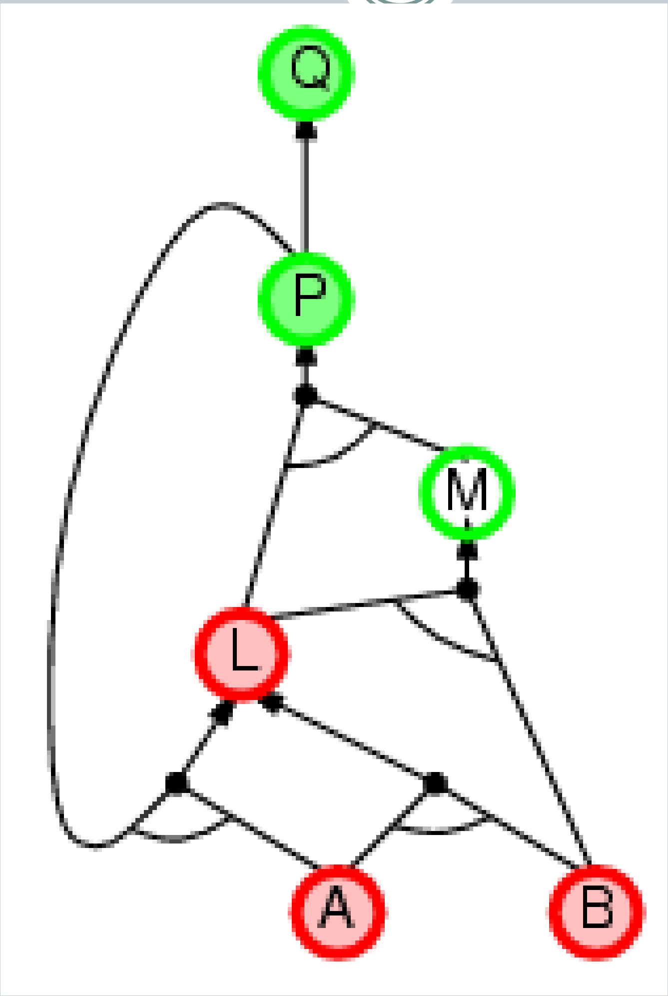


we need P to prove L and L to prove P.

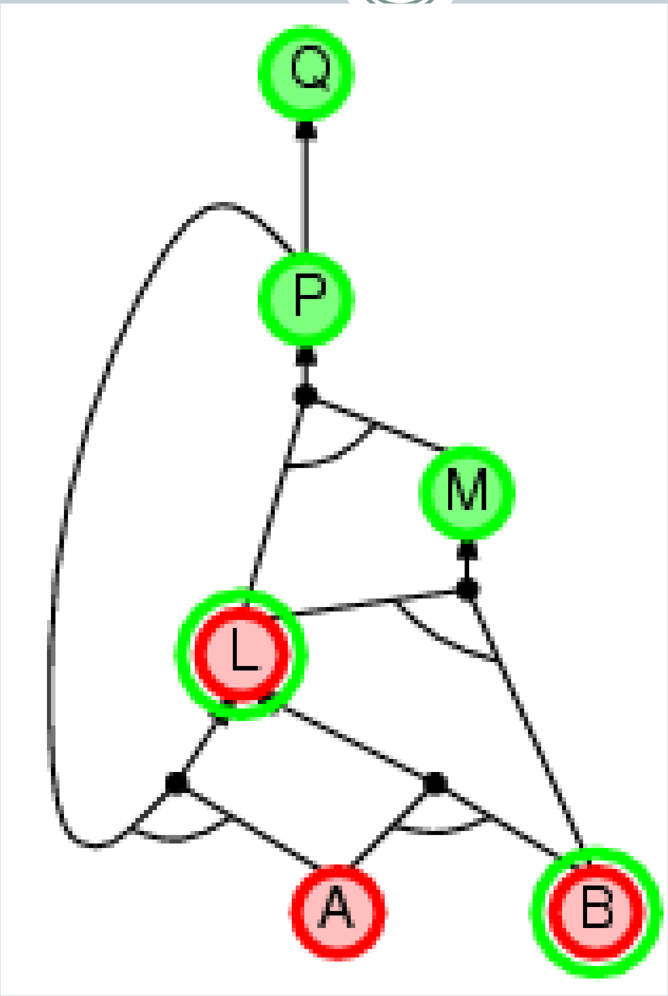
Backward chaining example



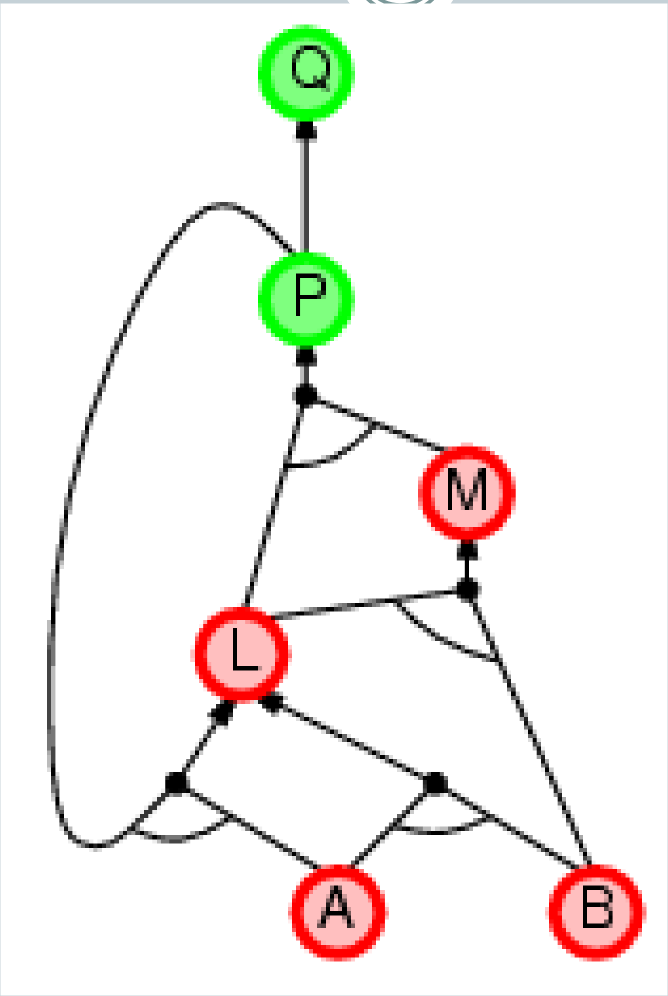
Backward chaining example



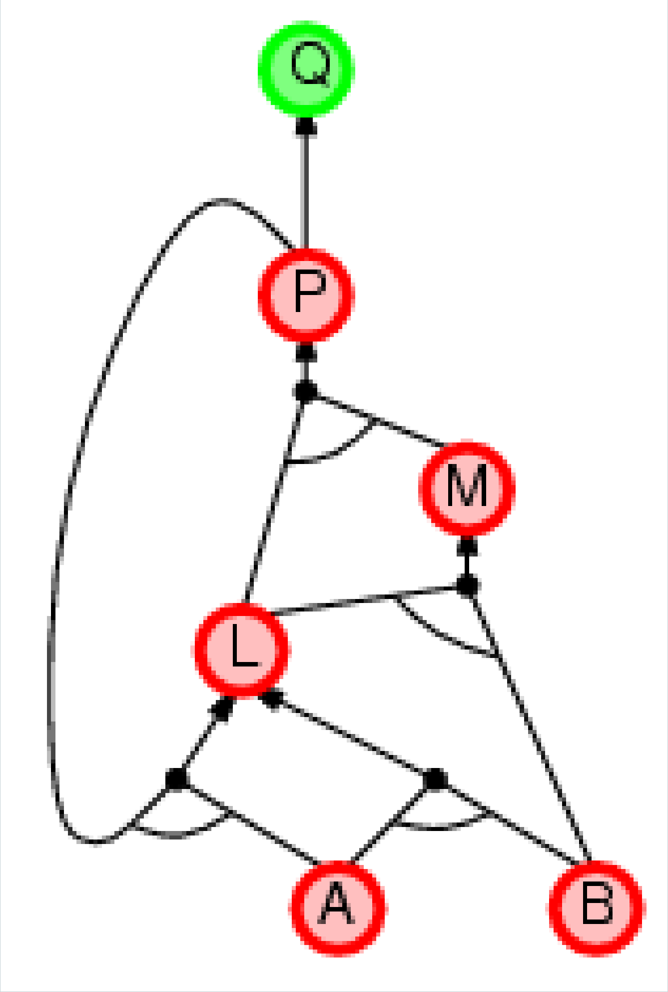
Backward chaining example



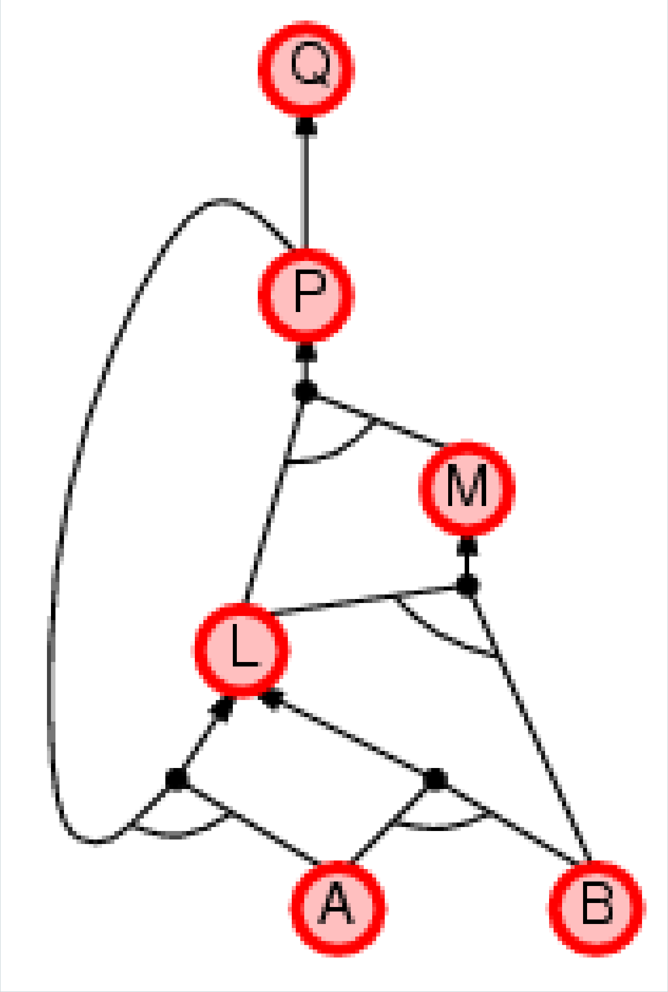
Backward chaining example



Backward chaining example



Backward chaining example



Backward chaining

- 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

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

Like FC, is linear and is also sound and complete (for Horn KB)

Model Checking

Two families of efficient algorithms:

- Complete backtracking search algorithms: DPLL algorithm
- Incomplete local search algorithms
  - `WalkSAT` algorithm

Satisfiability problems

- Consider *a* CNF sentence, e.g.,

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

  Satisfiability: Is there a model consistent with this sentence?

  $[A \vee B] \wedge [\neg B \vee \neg C] \wedge [A \vee C] \wedge [\neg D] \wedge [\neg D \vee \neg A]$

The `WalkSAT` algorithm

- Incomplete, local search algorithm
  - Begin with a random assignment of values to symbols
  - Each iteration: pick an unsatisfied clause
    - Flip the symbol that maximizes number of satisfied clauses, OR
    - Flip a symbol in the clause randomly
- Trades-off greediness and randomness
- Many variations of this idea
- If it returns failure (after some number of tries) we cannot tell whether the sentence is unsatisfiable or whether we have not searched long enough
  - If max-flips = infinity, and sentence is unsatisfiable, algorithm never terminates!
- Typically most useful when we expect a solution to exist

Pseudocode for WalkSAT

```
function WALKSAT(clauses, p, max-flips) returns a satisfying model or failure
   inputs: clauses, a set of clauses in propositional logic
           p, the probability of choosing to do a "random walk" move
           max-flips, number of flips allowed before giving up

   model ← a random assignment of true/false to the symbols in clauses
   for i = 1 to max-flips do
      if model satisfies clauses then return model
      clause ← a randomly selected clause from clauses that is false in model
      with probability p flip the value in model of a randomly selected symbol
          from clause
      else flip whichever symbol in clause maximizes the number of satisfied clauses
   return failure
```

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)

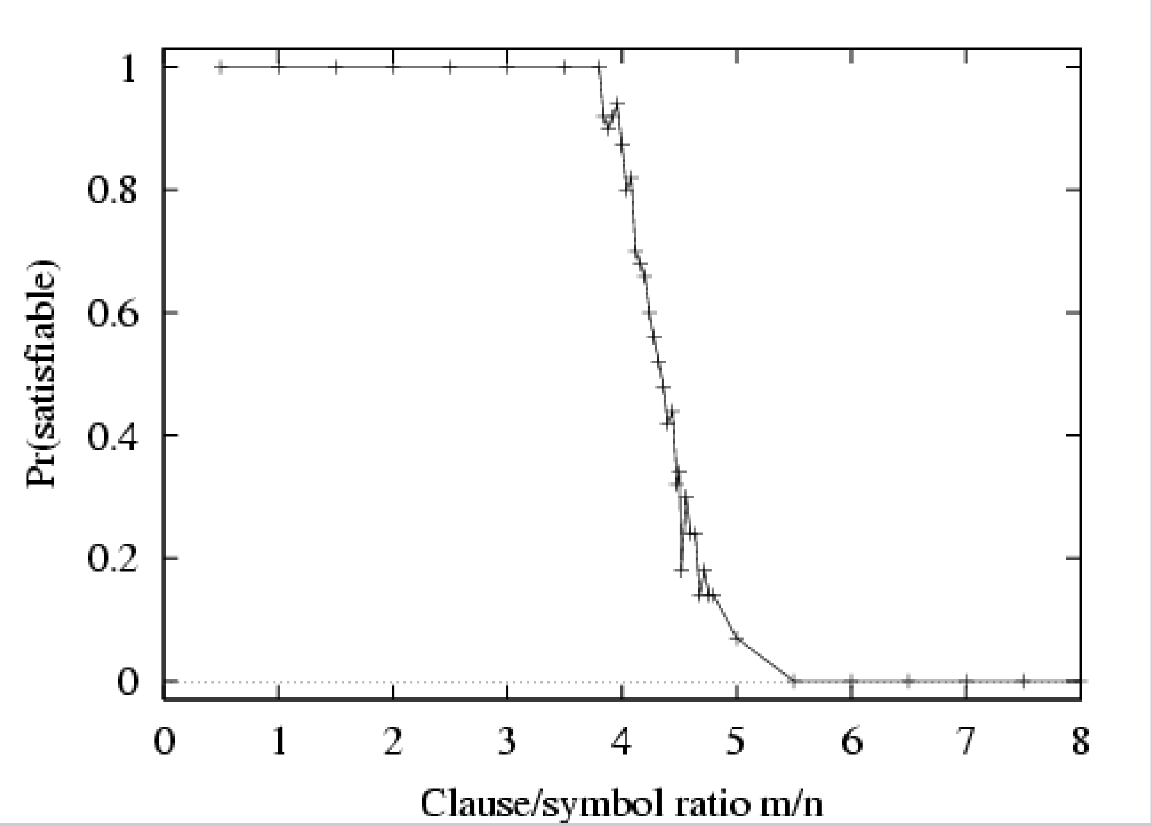
  n = number of symbols (5)

  - Underconstrained problems:
    - Relatively few clauses constraining the variables
    - Tend to be easy
    - 16 of 32 possible assignments above are solutions
      - (so 2 random guesses will work on average)

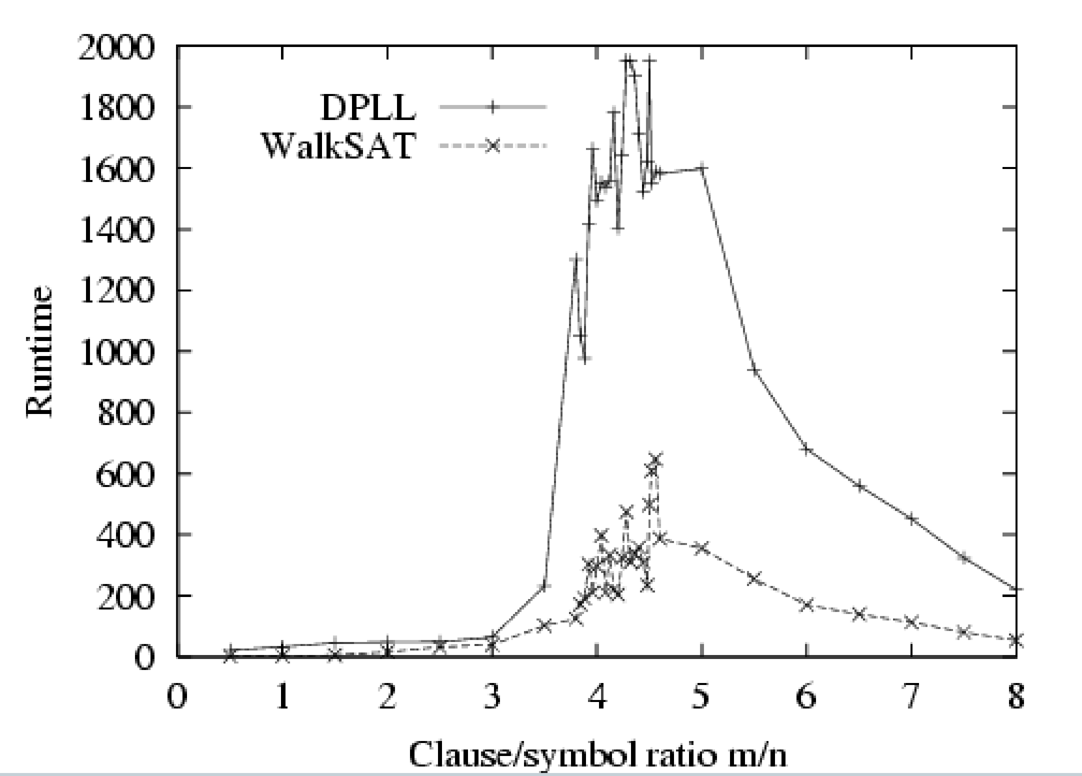
Hard satisfiability problems

- What makes a problem hard?
  - Increase the number of clauses while keeping the number of symbols fixed
  - Problem is more constrained, fewer solutions
  - Investigate experimentally….

P(satisfiable) for random 3-CNF sentences, n = 50



Run-time for DPLL and WalkSAT



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

Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

$\neg P_{1,1}$ (no pit in square [1,1])

$\neg W_{1,1}$ (no Wumpus in square [1,1])

$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$ (Breeze next to Pit)

$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$ (stench next to Wumpus)

$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$ (at least 1 Wumpus)

$\neg W_{1,1} \vee \neg W_{1,2}$ (at most 1 Wumpus)

$\neg W_{1,1} \vee \neg W_{8,9}$

…

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

Limited expressiveness of propositional logic

- KB contains "physics" sentences for every single square
- For every time *t* and every location [*x*,*y*],
  $L_{x,y} \wedge FacingRight^{t} \wedge Forward^{t} \Rightarrow L_{x+1,y}$
- Rapid proliferation of clauses.

First order logic is designed to deal with this through the introduction of variables.

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

- Propositional logic lacks expressive power