Logical Agents

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Spring 2016

Soleymani

"Artificial Intelligence: A Modern Approach", 3rd Edition, Chapter 7

Knowledge-based agents

- Knowledge-based agents
  - **Reasoning** operates on internal **representation of knowledge**
    - Logic as a general class of representation
      - Propositional logic
      - First-order logic

A generic knowledge-based agent

function KB_AGENT($percept$) **returns** an $action$
  persistent: KB, a knowledge base
    t, a counter for time, initially 0
  TELL(KB, MAKE_PERCEPT_SENTENCE($percept, t$))
  $action \leftarrow$ ASK(KB, MAKE_ACTION_QUERY(t))
  TELL(KB, MAKE_ACTION_SENTENCE($action, t$))
  $t \leftarrow t + 1$
return $action$

- Makes a sentence asserting that the agent perceived the given percept at the given time.
- Makes a sentence that asks what action should be done at the current time.
- Makes a sentence asserting that the chosen action was executed.

- The agent must be able to:
  - Represent states, actions, percepts,….
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

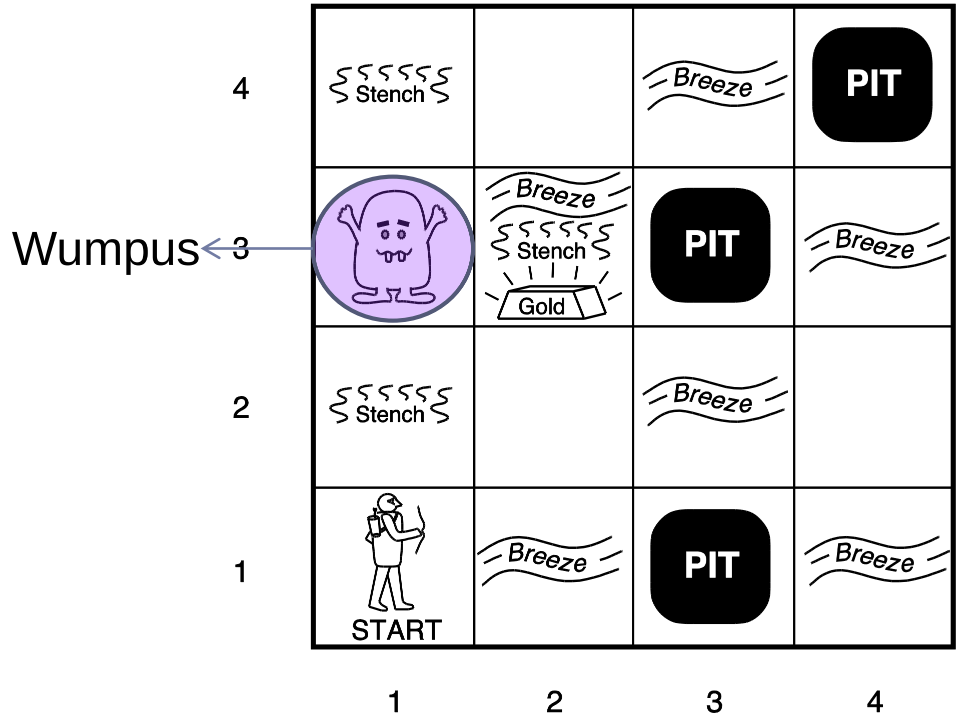
Knowledge Base (KB)

- KB = a set of sentences expressed in a knowledge representation language
  - TELL: adds new sentences to the knowledge base
  - ASK: asks a question of KB
    - the answer follows from previously TELLed sentences to the KB

- Inference: derives new sentences from old ones
  - Basis of TELL and ASK operations

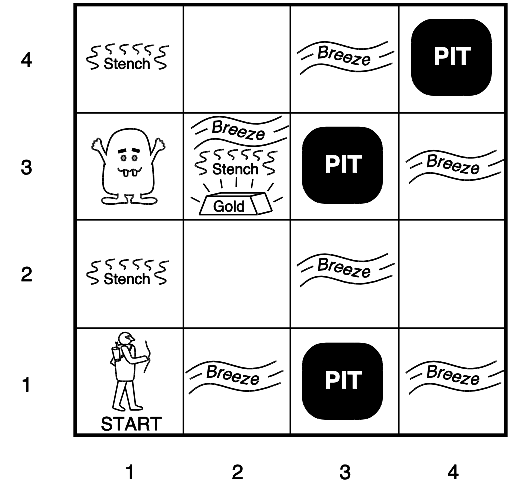
Wumpus world

- Wumpus
- Pitts
- Gold
- Agent



Wumpus world PEAS description

- Performance measure
  - +1000 for garbing gold
  - -1000 for death
  - -1 for each action
  - -10 for using up the arrow

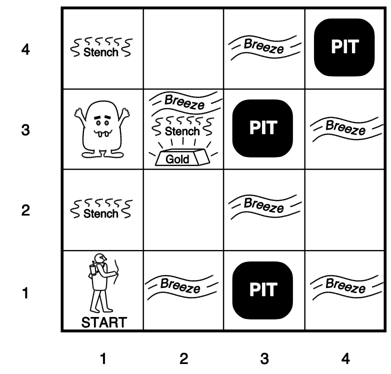


  - Game ends when the agent dies or climbs out of the cave

- Environment
  - 4×4 grid
  - Agent starts in [1,1] while facing to the right
  - Gold and wumpus are located randomly in the squares except to [1,1]
  - Each square other than [1,1] can be a pit with probability 0.2

Wumpus world PEAS description

- Sensors: Stench, Breeze, Glitter, Bump, Scream
  - In the squares adjacent to wumpus, agent perceives a Stench
  - It the squares adjacent to a pit, agent perceives a Breeze
  - In the gold square, agent perceives a Glitter
  - When walking into a wall, agent perceives a Bump
  - When Wumpus is killed, agent perceives a Scream



- Actuators: Forward, TurnLeft, TurnRight, Shoot, Grab, Climb
  - Forward, TurnLeft, TurnRight: moving and rotating face actions.
    - Moving to a square containing a pit or a live wumpus causes death.
    - If an agent tries to move forward and bumps into a wall then it does not move.
  - Shoot: to fire an arrow in a straight line in the facing direction of the agent
    - Shooting kills wumpus if the agent is facing it (o.w. the arrow hits a wall)
    - The first shoot action has any effect (the agent has only one arrow)
  - Grab: to pick up the gold if it is in the same square as the agent.
  - Climb: climb out of the cave but only from [1,1]

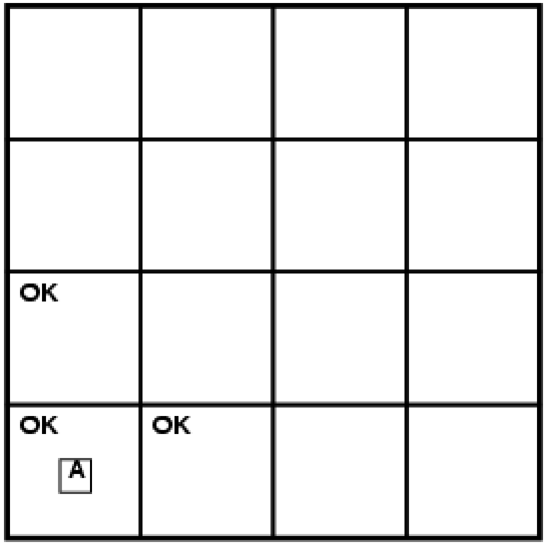
Wumpus world characterization

- Fully Observable? No – many aspects are not directly perceivable
- Episodic? No – sequential at the level of actions
- Static? Yes
- Discrete? Yes
- Single-agent? Yes

Wumpus world example

A: agent

OK: safe square

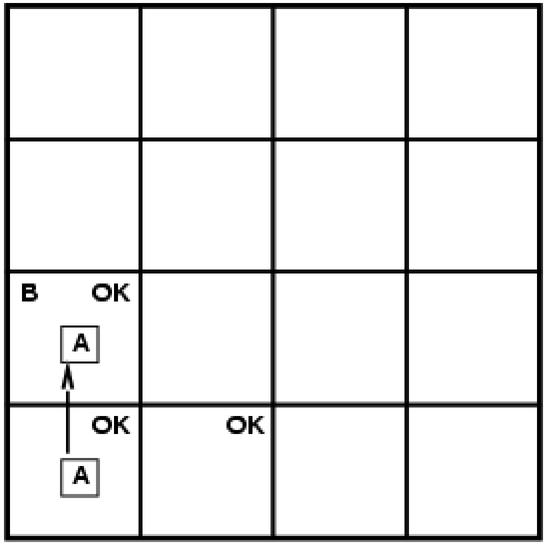


Wumpus world example

A: agent

OK: safe square

B: Breeze



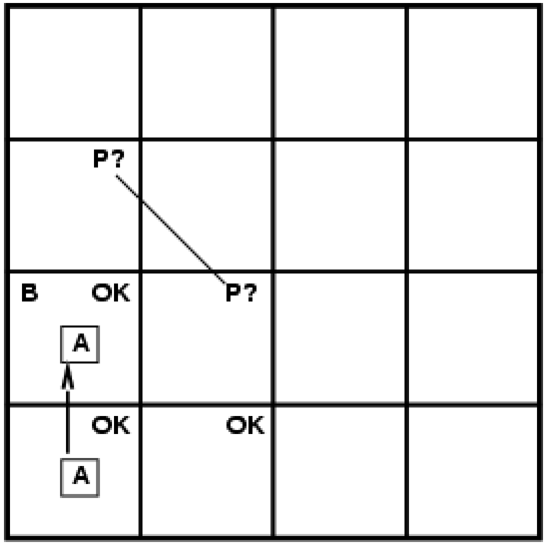
Wumpus world example

A: agent

OK: safe square

B: Breeze

P: Pit



Wumpus world example

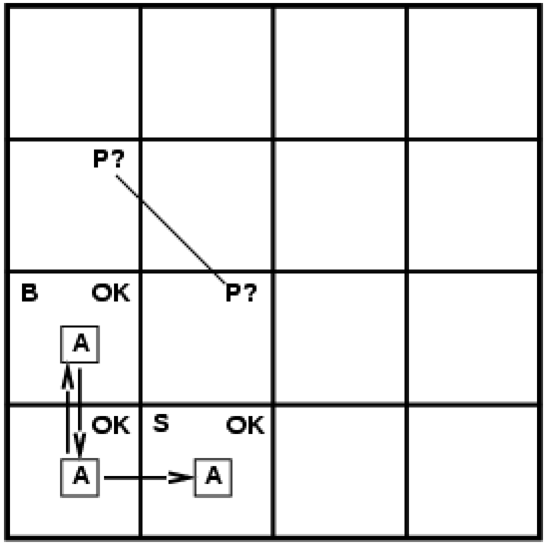
A: agent

OK: safe square

B: Breeze

P: Pit

S: Stench



Wumpus world example

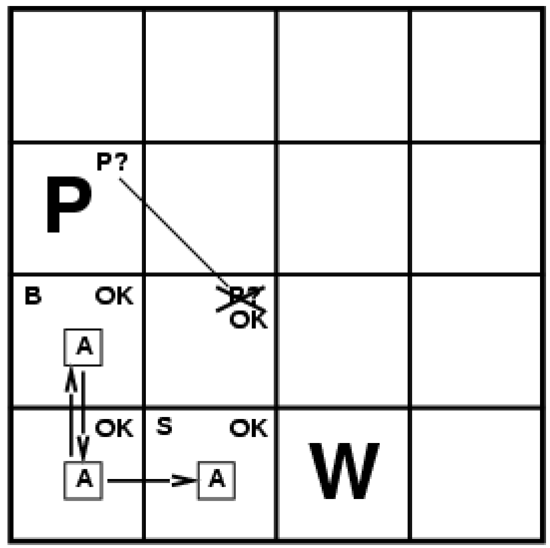
A: agent

OK: safe square

B: Breeze

P: Pit

S: Stench



Wumpus world example

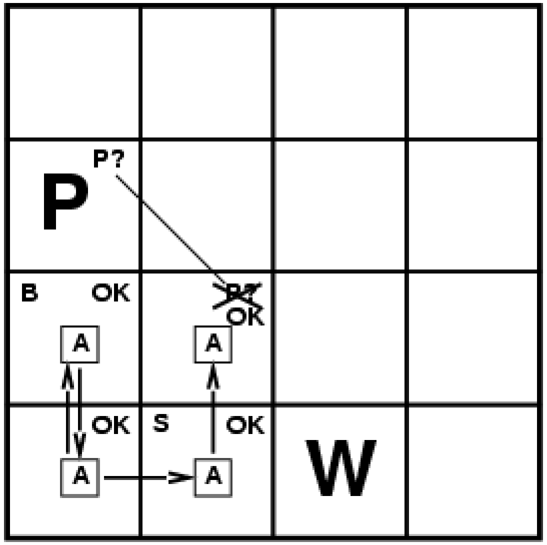
A: agent

OK: safe square

B: Breeze

P: Pit

S: Stench



Wumpus world example

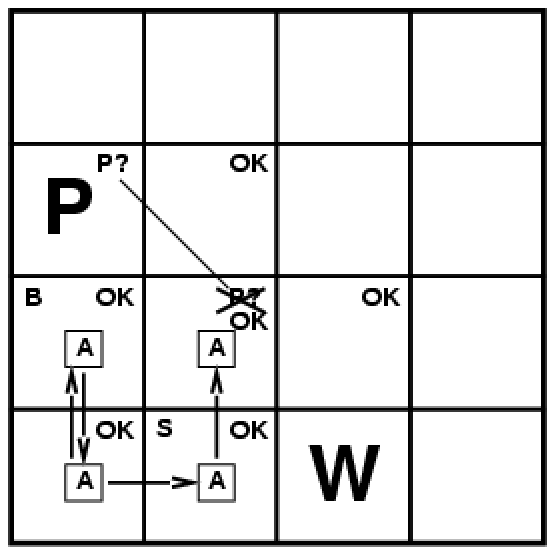
A: agent

OK: safe square

B: Breeze

P: Pit

S: Stench



Wumpus world example

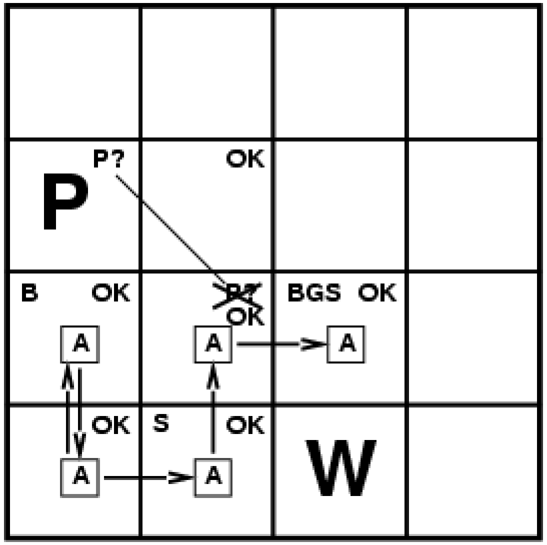
A: agent

OK: safe square

B: Breeze

P: Pit

S: Stench



Logic & Language

- Logic includes formal languages for representing information such that conclusions can be drawn
- Syntax defines the well-formed sentences in a language
- Semantics define the "meaning" of sentences
  - i.e., define truth of a sentence with respect to each possible world

- We will introduce Propositional Logic in this lecture
  - It talks about facts
  - Propositions can be true, false, or unknown

Propositional logic: Syntax

- Propositional logic is the simplest logic
  - To illustrate basic ideas about logic & reasoning

- The proposition symbols $P, Q, \ldots$ are sentences.
- If P is a sentence, $\neg P$ is a sentence (negation)
- If P and Q are sentences, $P \wedge Q$ is a sentence (conjunction)
- If P and Q are sentences, $P \vee Q$ is a sentence (disjunction)
- If P and Q are sentences, $P \Rightarrow Q$ is a sentence (implication)
- If P and Q are sentences, $P \Leftrightarrow Q$ is a sentence (biconditional)

Propositional logic (Grammar)

$$
\begin{aligned}
Sentence &\rightarrow AtomicSentence \mid ComplexSentence \\
AtomicSentence &\rightarrow True \mid False \mid P \mid Q \mid R \mid \ldots \\
ComplexSentence &\rightarrow (Sentence) \\
&\mid \neg Sentence \\
&\mid Sentence \wedge Sentence \\
&\mid Sentence \vee Sentence \\
&\mid Sentence \Rightarrow Sentence \\
&\mid Sentence \Leftrightarrow Sentence
\end{aligned}
$$

$Precedence: \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Truth tables for connectives

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

Propositional logic: Semantics

- Each model specifies true/false of each proposition symbol
  - e.g., Propistion symbols: $P_{1,2}, P_{2,2}, \ P_{3,1}$
    - 8 possible models
    - $m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\}$

- Semantics of a complex sentence in any model *m*:
  - $\neg P$ is true iff P is false in m.
  - $P \wedge Q$ is true iff P is true and Q is true in m.
  - $P \vee Q$ is true iff P is true or Q is true in m.
  - $P \Longrightarrow Q$ is true unless P is true and Q is false in m.
  - $P \Leftrightarrow Q$ is true iff P and Q are both true or both false in m.

Wumpus world sentences

- $P_{i,j}$ is true if there is a pit in $[i,j]$.
- $W_{i,j}$ is true if there is a wumpus in $[i,j]$, dead or alive.
- $B_{i,j}$ is true if the agent perceives a breeze in $[i,j]$.
- $S_{i,j}$ is true if the agent perceives a stench in $[i,j]$.

- General rules (only related ones to the current agent position)
  - $R_1: \neg P_{1,1}$ (no pit in [1,1])
  - $R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ (Pits cause breezes in adjacent squares)
  - $R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$ (Pits cause breezes in adjacent squares)

- Perception
  - $R_4: \neg B_{1,1}$
  - $R_5: B_{2,1}$

Models

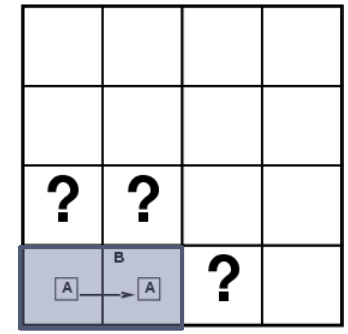
- Models are <u>mathematical abstraction</u> of possible worlds
  - Each model fixes the truth or falsehood of every relevant sentence.

- We can consider a set for each logical sentence that specifies all possible worlds in which α is true

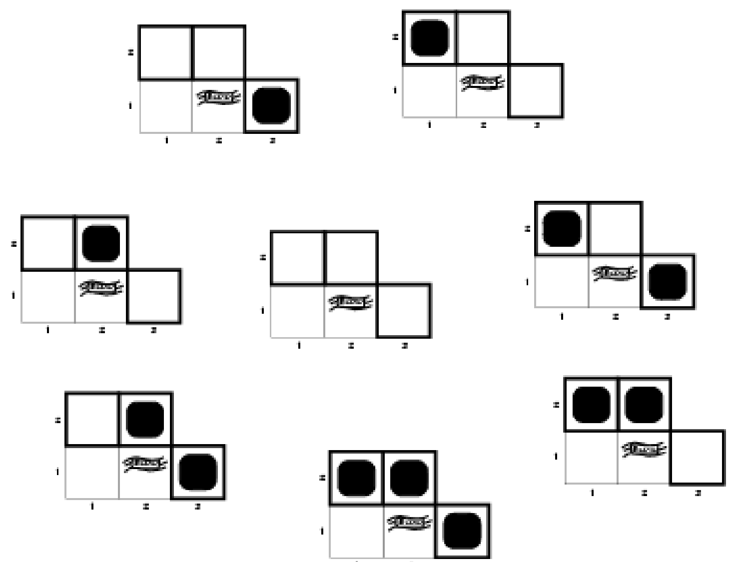
- $M(\alpha)$: set of all models of the sentence α (all satisfying α)
  - $m \in M(\alpha)$ if α is true in model m
  - E.g., For α: $x + y = 4$, all possible assignments of values to x, y are models. $M(\alpha)$ contains a subset of models satisfying $x + y = 4$.

Wumpus models

- Agents starts at [1,1] with no active sensor, then moves to [2,1] and senses Breeze in it



- Possible models for pits in [1,2], [2,2], [3,1]:



Logical reasoning: entailment

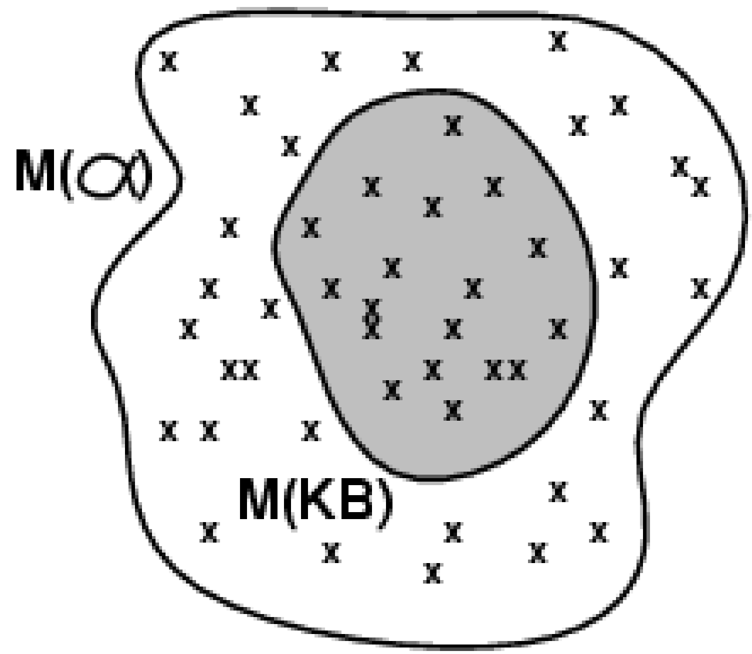
- $\alpha \models \beta$ (entailment): α entails β
  - $\alpha \Rightarrow \beta$ **is a tautology or valid**
    - A sentence is **valid** or **tautology** if it is $True$ in all models (e.g., $P \vee \neg P$, $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$)
  - β logically follows from α or α is stronger than β

- $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$
  - α entails β iff β is true in all worlds where α is true

Entailment

- $KB \vDash \alpha$ iff $M(KB) \subseteq M(\alpha)$

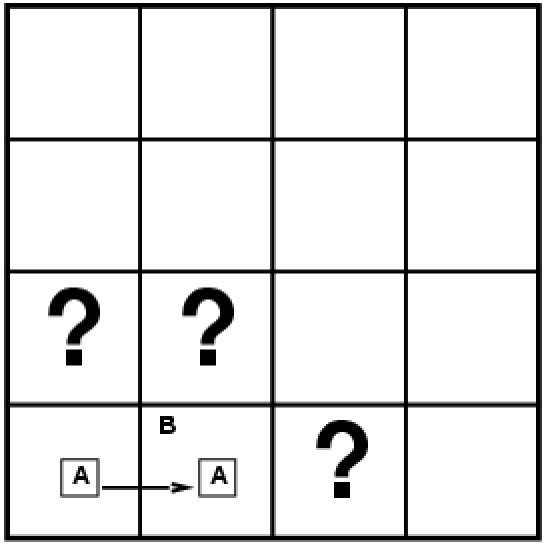
- Example:
  - *KB*="A is red" & "B is blue"
  - α = "A is red"



Entailment in the wumpus world

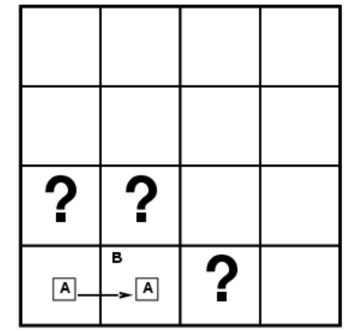
- KB (before perception): rules of the wumpus world

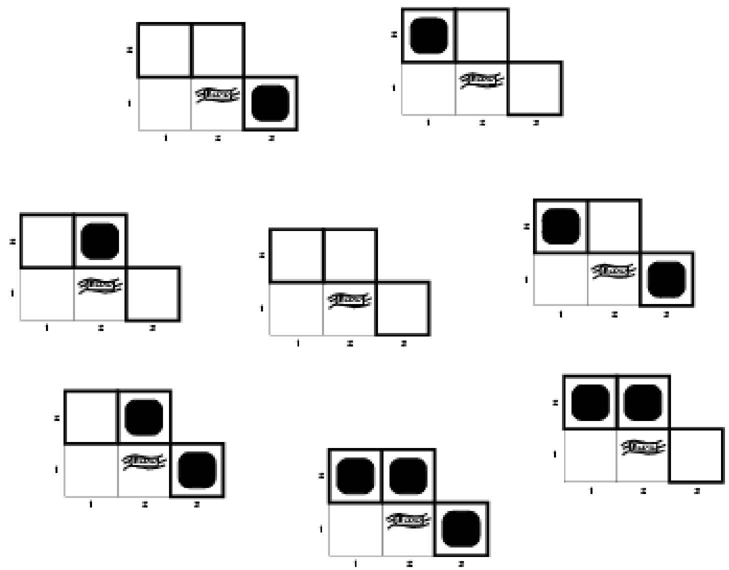
- Perception: After detecting nothing in [1,1], moving right, a breeze in [1,2]



Wumpus models

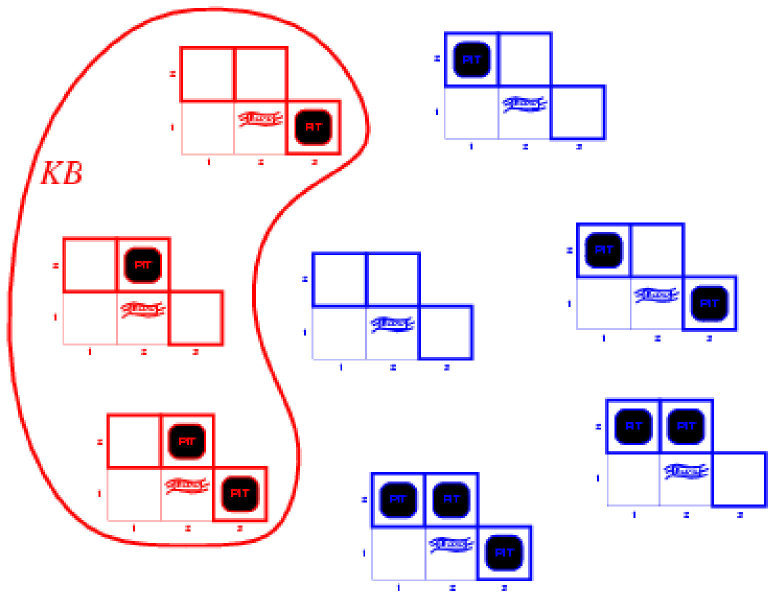
- Possible models for pits in [1,2], [2,2], [3,1]

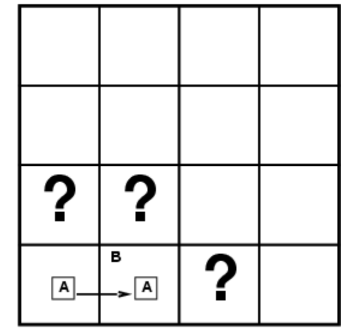




Consider possible models for only pits in neighboring squares $2^3 = 8$ possible models

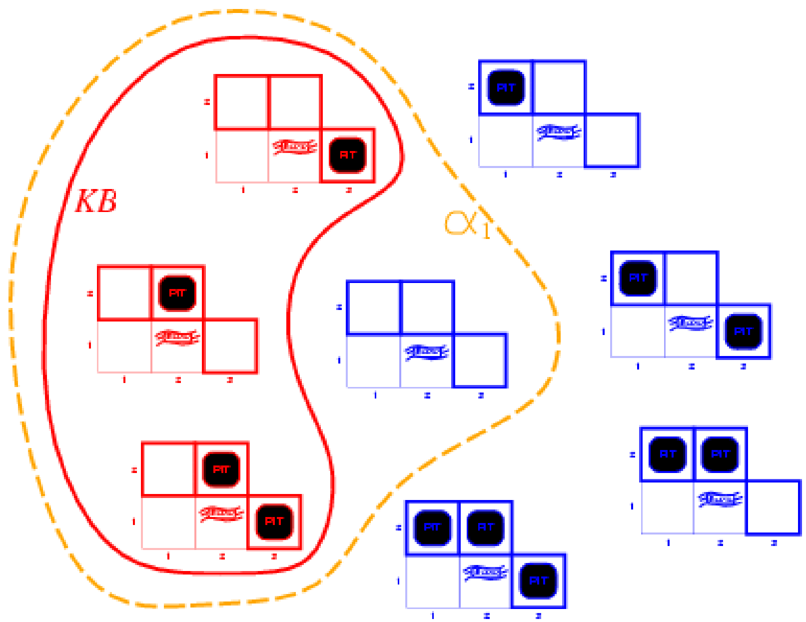
Wumpus models

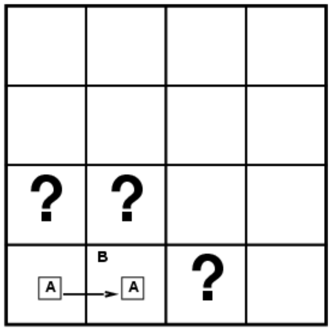




- KB = wumpus-world rules + perceptions

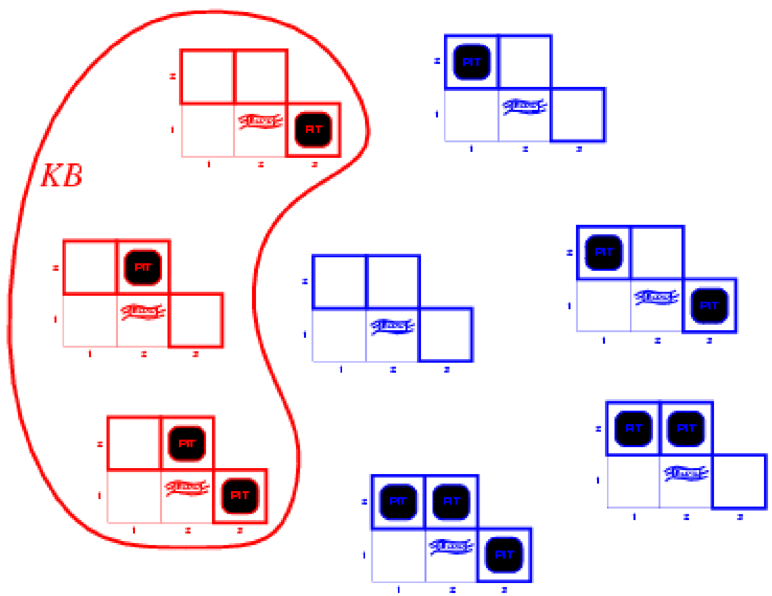
Wumpus models

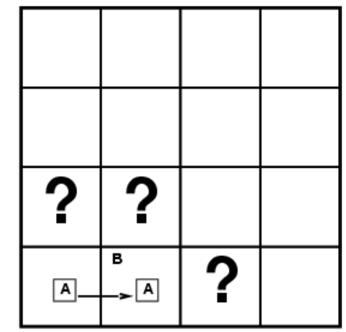




- KB = wumpus-world rules + perceptions
- α_1 = "[1,2] is safe"
  - $M(KB) \subseteq M(\alpha_1) \Rightarrow KB \vDash \alpha_1$

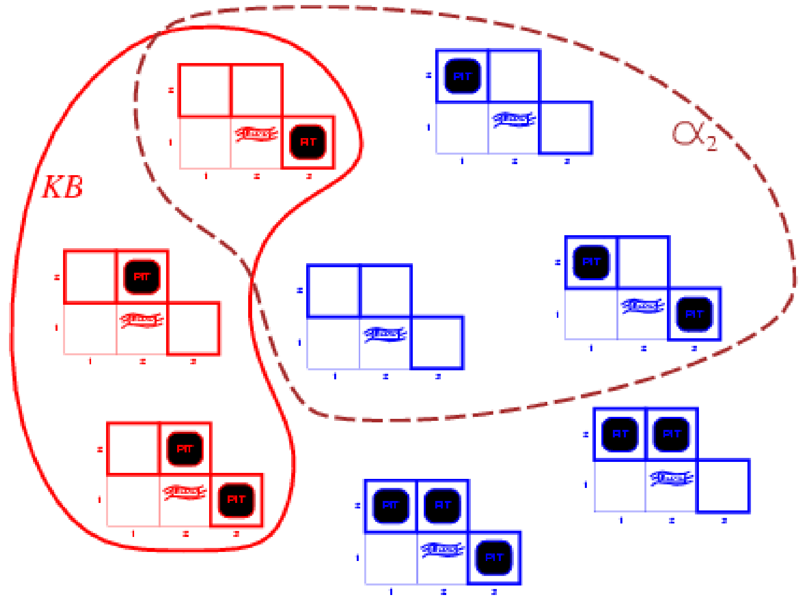
Wumpus models

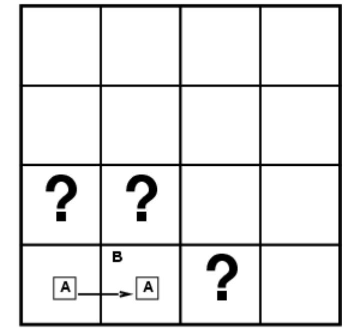




- *KB* = wumpus-world rules + perceptions

Wumpus models





- KB = wumpus-world rules + perceptions
- α_2 = "[2,2] is safe", $KB \nvDash \alpha_2$

Inference

- $KB \vdash_i \alpha$: α can be derived from KB by an inference algorithm i

- An inference algorithm i is:
  - sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \vDash \alpha$
    - all provable statements by this algorithm are true
  - complete if whenever $KB \vDash \alpha$, it is also true that $KB \vdash_i \alpha$
    - all true statements are provable by this algorithm

Inference methods

- Inference methods can be categorized into:
  - Model checking (enumerating models)
    - truth table enumeration (always exponential in n)
    - improved backtracking, e.g., Davis-Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms
  - Theorem proving (searching proofs by applying inference rules)
    - Applying a sequence of <u>inference rules</u> on KB to find the desired sentence
    - Legitimate (sound) generation of new sentences from old ones.

Truth tables for inference

- $KB \vDash \alpha$?
  - Enumerate the models
  - "Is α true in every model in which KB is true?"

KB

R_1: $\neg P_{1,1}$
R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
R_4: $\neg B_{1,1}$
R_5: $B_{2,1}$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Entailment by enumeration (model checking)

function $TT\_ENTAILS?(KB, \alpha)$ **returns** $true$ or $false$

inputs: KB, the knowledge base, 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 $TT\_CHECK\_ALL(KB, \alpha, symbols, \{\})$

---

function $TT\_CHECK\_ALL(KB, \alpha, symbols, model)$ **returns** $true$ or $false$

if $EMPTY?(symbols)$ **then**

if $PL\_TRUE?(KB, model)$ **then return** $PL\_TRUE?(\alpha, model)$

else return $true$

else

$P \leftarrow FIRST(symbols)$

$rest \leftarrow REST(symbols)$

return $TT\_CHECK\_ALL(KB, \alpha, rest, model \cup \{P = true\})$ **and**

$TT\_CHECK\_ALL(KB, \alpha, rest, model \cup \{P = false\})$

Entailment by enumeration: properties

- Depth-first enumeration of all models is sound and complete (when finite models).

- For n symbols, time complexity is $O(2^n)$, space complexity is $O(n)$.

Satisfiability

- A sentence is satisfiable if it is true in some models
  - e.g., $P \lor Q$

- Determining the satisfiability of sentences in propositional logic (SAT problem) is NP-complete

- **Satisfiability** and **validity** are connected.
  - α is valid or tautology iff $\neg\alpha$ is unsatisfiable.
  - α is satisfiable iff $\neg\alpha$ is not valid.

$$\alpha \vDash \beta \text{ iff } (\alpha \land \neg\beta) \textit{ is unsatisfiable}$$

Proof by contradiction

The **DPLL** (Davis, Putnam, Logemann, Loveland) algorithm

- Improvements over truth table enumeration (simple DFS):
  - Early termination
    - A clause is true if any literal is true.
    - A sentence is false if any clause is false.
  - Pure symbol heuristic
    - Pure symbol: always appears with the same "sign" in all clauses.
      - e.g., In $(A \vee \neg B) \wedge (\neg B \vee \neg C) \wedge (C \vee A)$, A and B are pure, C is impure.
    - Make a pure symbol literal true.
    - In determining purity, we can ignore clauses already known to be true
  - Unit clause heuristic
    - Unit clause: all literals except to one already assigned false by model.
      - The only literal in a unit clause will be assigned true.

- Many tricks can be used to enable SAT solvers to scale up.

The DPLL algorithm

Determining satisfiability of an input propositional logic sentence (in CNF)

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(*clauses*, *symbols*, [ ])

---

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

  if every clause in *clauses* is true in *model* **then return** *true*
  if some clause in *clauses* is false in *model* **then return** *false*
  P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols*–P, [P = *value*|*model*])
  P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols*–P, [P = *value*|*model*])
  P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  return DPLL(*clauses*, *rest*, [P = *true*|*model*]) **or**
      DPLL(*clauses*, *rest*, [P = *false*|*model*])

Entailment by using inference rule

- Propositional theorem proving

- Searching proofs can be more efficient than model checking.
  - Can ignore irrelevant propositions.
  - When the number of models is large but the length of proof is short, entailment by theorem proving is useful.

Inference rules

- Some samples:
  - Modus Ponens: $\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$
  - And elimination: $\frac{\alpha \wedge \beta}{\beta}$
  - Biconditional elimination: $\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$

Valid implication

- Deduction theorem: $\alpha \vDash \beta$ iff $\alpha \Rightarrow \beta$ is valid.
  - Every **valid implication** describes a legitimate inference.

- Modus Ponens: $\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$ is equivalent to valid implication
  - $\big((\alpha \Rightarrow \beta) \land \alpha\big) \Rightarrow \beta$ is valid

- All logical entailments can be used as inference rules.

Logical equivalence

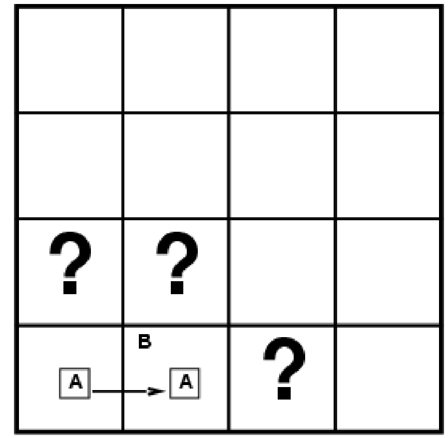
- Using logical equivalence we can find many inference rules
- Two sentences are logically equivalent iff they are true in the same models

$$\alpha \equiv \beta \text{ iff } \alpha \vDash \beta \text{ and } \beta \vDash \alpha$$

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

Example of inference

- R_1: $\neg P_{1,1}$
- R_2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- R_3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- R_4: $\neg B_{1,1}$
- R_5: $B_{2,1}$



- Can we infer from R_1 through R_5 to prove $\neg P_{1,2}$?
  - R_6: $\left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})\right) \wedge \left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}\right)$ [biconditional elim. to R_2]
  - R_7: $\left((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}\right)$ [and elim. to R_6]
  - R_8: $\left(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})\right)$ [$(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$]
  - R_9: $\neg(P_{1,2} \vee P_{2,1})$ [Modus Ponens with R_8 and R_4]
  - R_{10}: $\neg P_{1,2} \wedge \neg P_{2,1}$ [De Morgan' rule]

Finding a proof as a search problem

- *INITIAL STATE*: the initial knowledge base
- *ACTIONS*: all inference rules applied to all the sentences that match their top line
- *RESULT*: add the sentence in the bottom line of the used inference rule to the current ones
- *GOAL*: a state containing the sentence we are trying to prove.

Monotonicity property

- The set of entailed sentences can only grow as information is added to KB.
  - Inference rules can be applied where premises are found in KB (regardless of what else is in KB)

- If $KB \vDash \alpha$ then $KB \wedge \beta \vDash \alpha$ (for any α and β)
  - The assertion β can not invalidate any conclusion α already inferred.

Resolution

- Inference rules are sound.
- Is a set of rules complete?
  - Are the available inference rules adequate?

- Resolution: a single inference rule that yields a complete inference algorithm (when couples with any complete search algorithm).

Resolution

- Unit resolution: l_i and m are complementary literals

$$\frac{l_1 \vee l_2 \vee \cdots \vee l_k, \qquad m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k}$$

- Full resolution rule: l_i and m_j are complementary literals

$$\frac{l_1 \vee l_2 \vee \cdots \vee l_k, \qquad m_1 \vee m_2 \vee \cdots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

- It resolution rule sound. Why?
- A resolution-based theorem prover can (for any sentences α and β in propositional logic) decide whether $\alpha \vDash \beta$
  - Is $\alpha \wedge \neg\beta$ unsatisfiable?
  - CNF (Conjunctive Normal Form)

CNF Grammar

$$CNFSentence \rightarrow Clause_1 \wedge \cdots \wedge Clause_n$$

$$Clause \rightarrow Literal_1 \vee \cdots \vee Literal_m$$

$$Literal \rightarrow Symbol \mid \neg Symbol$$

$$Symbol \rightarrow P \mid Q \mid R \mid \ldots$$

Example of resolution

$$\frac{P_{1,1} \vee P_{3,1}, \qquad \neg P_{1,1} \vee \neg P_{2,2}}{P_{3,1} \vee \neg P_{2,2}}$$

Resolution algorithm

- To show $KB \vDash \alpha$, we show $KB \wedge \neg\alpha$ is unsatisfiable

function PL-RESOLUTION(KB,α) **returns** $true$ or $false$
input: KB, the knowledge base (a sentence in propositional logic)
α, the query (a sentence in propositional logic)

$clauses$ ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
new ← {}
loop do
  for each pair of clauses C_i, C_j **in** $clauses$ **do**
    $resolvents \leftarrow PL\_RESOLVE\ (Ci, C_j)$
    if $resolvents$ contains the empty clause **then return** $true$
    $new \leftarrow new \cup resolvents$
  if $new \subseteq Clauses$ **then return** $false$
  $clauses \leftarrow clauses \cup new$

Resolution algorithm

- To show $KB \vDash \alpha$, we show $KB \wedge \neg\alpha$ is unsatisfiable

function PL-RESOLUTION(KB,α) **returns** $true$ or $false$
input: KB, the knowledge base (a sentence in propositional logic)
α, the query (a sentence in propositional logic)

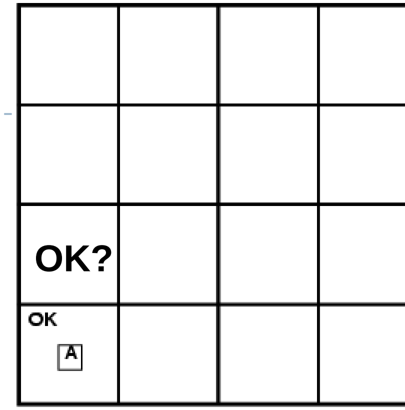
$clauses$ ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
new ← {}
loop do

Each pair containing complementary literals is resolved and the resulted clause is added to the set if it is not already present.

The process continues until one of these happens:
- No new clauses ($KB \nvDash \alpha$)
- Two clauses resolve to yield the empty clause ($KB \vDash \alpha$)

Sound and complete inference algorithm

Resolution example

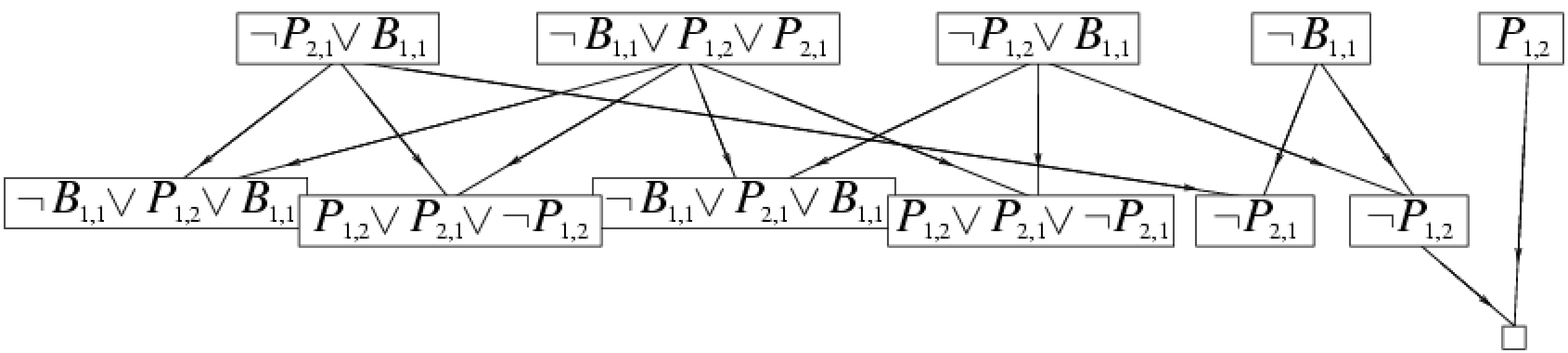


$$KB = \left(B_{1,1} \Leftrightarrow \left(P_{1,2} \vee P_{2,1}\right)\right) \wedge \neg B_{1,1}$$

$$\alpha = \neg P_{1,2}$$

- Convert $KB \wedge \neg\alpha$ into CNF

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



Contradiction

Completeness of resolution

- Resolution closure $RC(S)$: a set of clauses derivable by repeated application of resolution rule to clauses in S or their derivatives.

- $RC(S)$ is finite. Thus PL_onstructed out of $P_1, P_2, \ldots, P_k$ that appear in S. $RESOLUTION$ terminates.
  - Finite distinct clauses can be c

- Ground resolution theorem: If a set of clauses S is unsatisfiable then $RC(S)$ contains the empty clause.
  - If $RC(S)$ does not contain the empty clause, we can find an assignment of values to the symbols that satisfies $RC(S)$ and thus S

Horn clauses & definite clauses

- Some real-world KBs satisfy restrictions on the form of sentences
  - We can use more restricted and efficient inference algorithms

- Definite clause: a disjunction of literals of which exactly one is positive literal.

$$\neg P_1 \vee \neg P_2 \vee \ldots \vee \neg P_k \vee P_{k+1} \Leftrightarrow (P_1 \wedge P_2 \wedge \ldots \wedge P_k) \Rightarrow P_{k+1}$$

- Horn clause: a disjunction of literals of which at most one is positive literal.
  - Closed under resolution

Grammar

$$Symbol \rightarrow P \mid Q \mid R \mid \ldots$$

$$HornClause \rightarrow DefiniteClause \mid GoalClause$$

$$DefiniteClause \rightarrow (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow Symbol$$

$$GoalClause \rightarrow (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow False$$

Forward and backward chaining

- Inference with Horn clauses can be done through forward chaining or backward chaining.

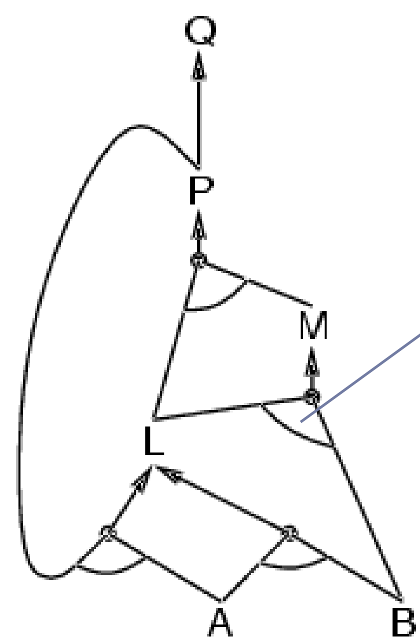
- These algorithms are very natural and run in time that is linear in the size of KB.

- They are bases of logic programming
  - Prolog: backward chaining

Forward chaining

- Repeated application of Modus Ponens until reaching goal
- Fire any rule whose premises are satisfied in the *KB*
  - add its conclusion to the *KB*, until query is found

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



Arcs show conjunctions

Forward chaining algorithm

function $PL\_FC\_ENTAILS?(KB, q)$ **returns** $true$ or $false$
inputs: KB, the knowledge base (a set of propositional definite clauses)
q, the query (a propositional symbol)
$count \leftarrow$ a table where $count[c]$ is the number of symbols in c's premise
$inferred \leftarrow$ a table, where $inferred[s]$ is initially false for all symbols
$agenda \leftarrow$ a queue of symbols, initially symbols known to be true in KB

while $agenda \neq \{\}$ **do**
$p \leftarrow POP(agenda)$
if $p = q$ **then return** $true$
if $inferred[p] = false$ **then**
$inferred[p] \leftarrow true$
for each clause c in KB where p is in $c.PREMISE$ **do**
$count[c]$= $count[c]$-1
if $count[c] = 0$ **then** add $c.CONCLUSION$ to $agenda$
return false

Begin with known facts (positive literals) in KB

The process continues until q is added or no further inference

Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

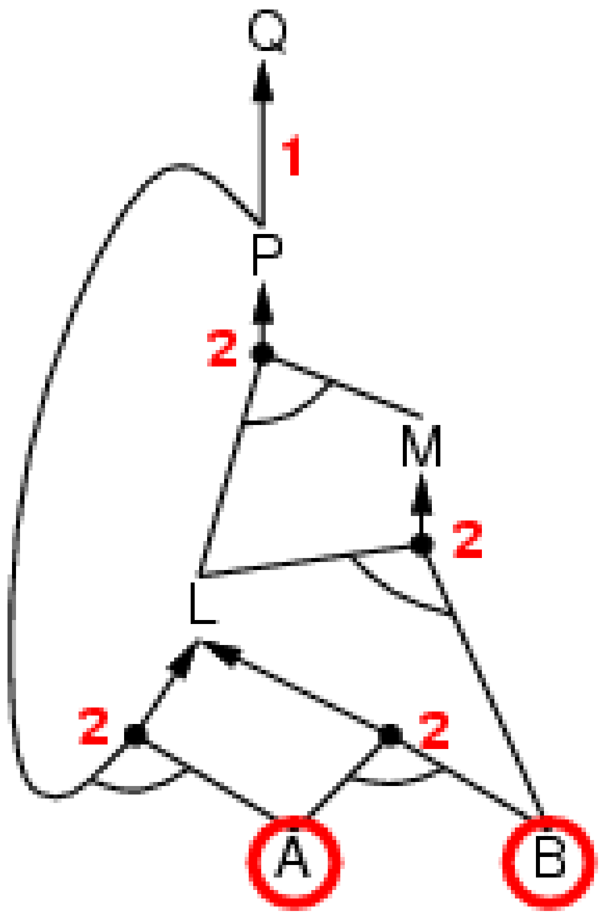
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

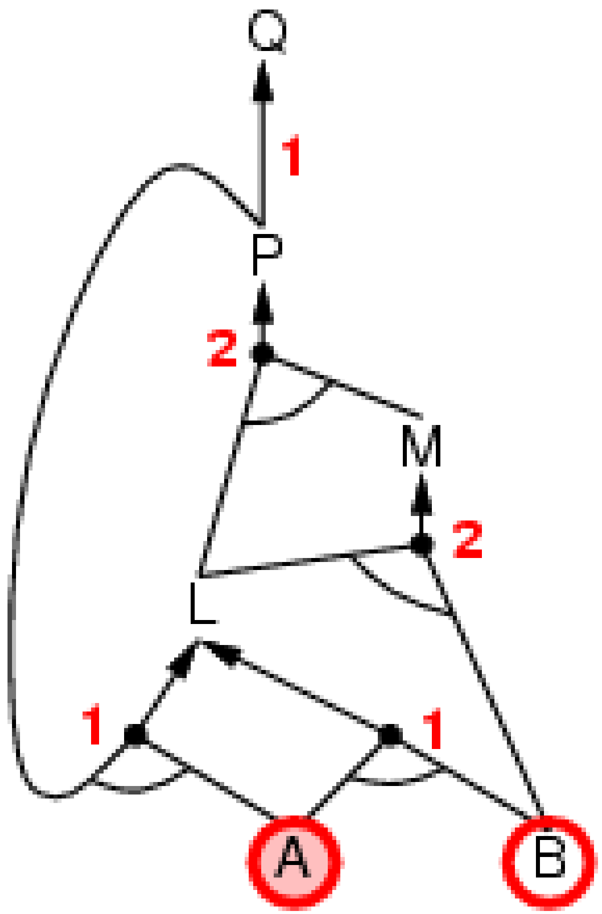
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

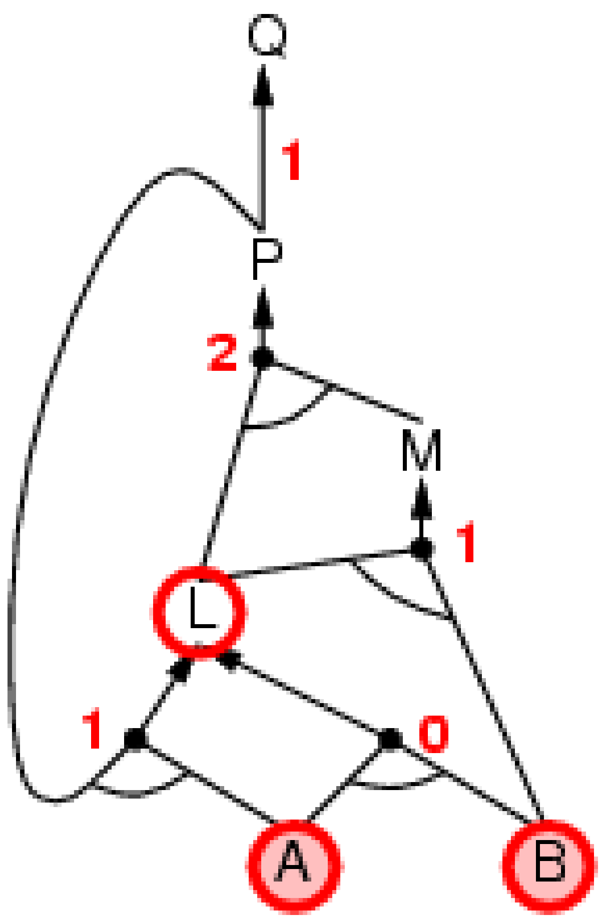
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

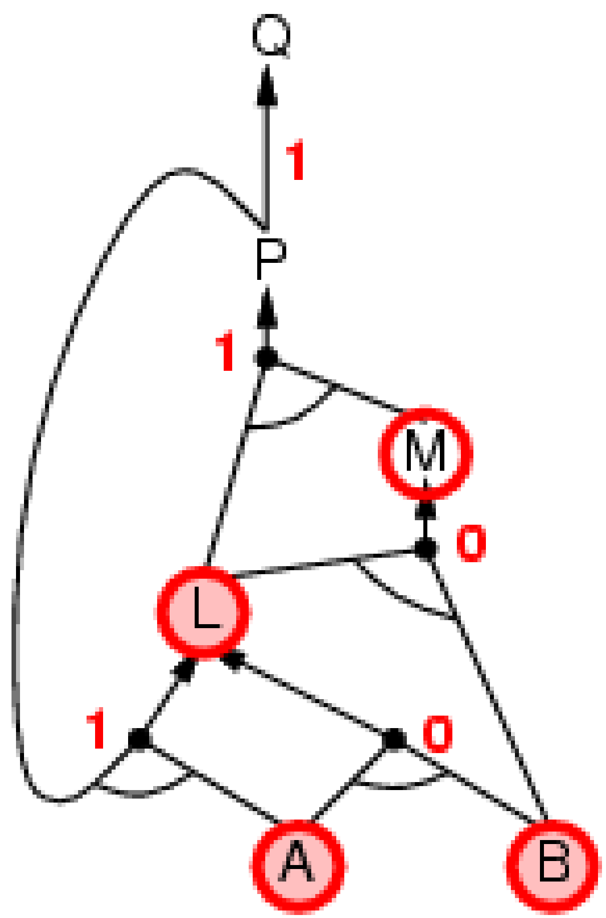
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

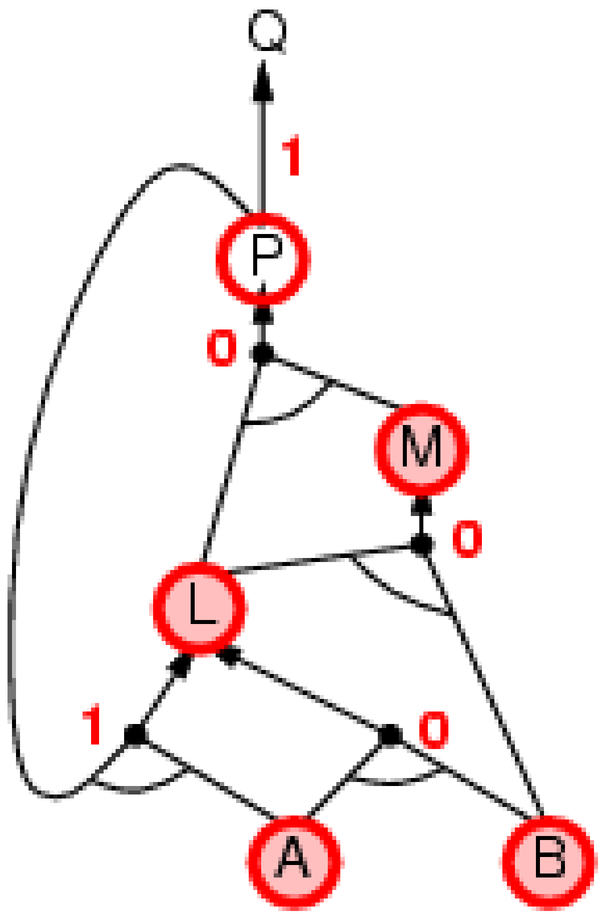
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

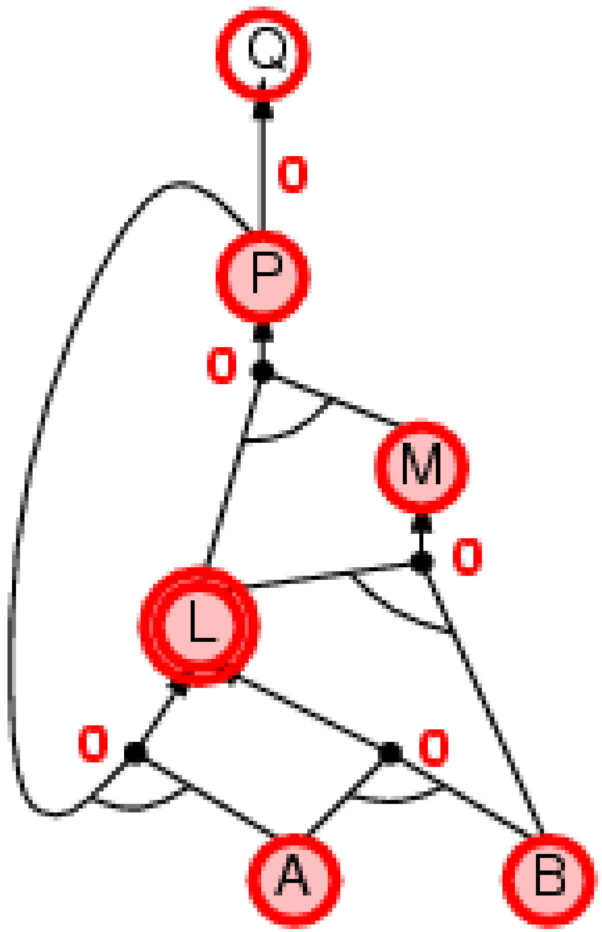
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

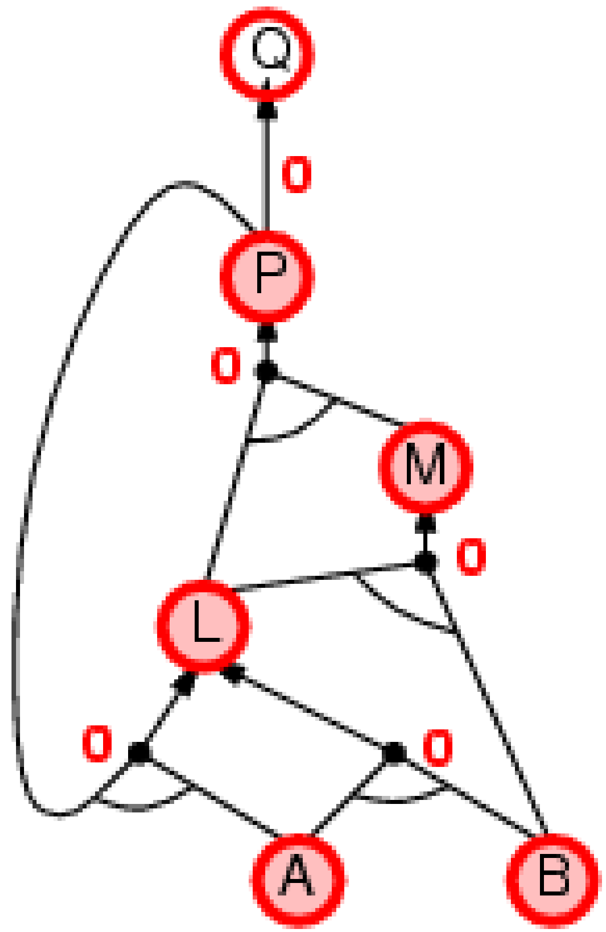
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining: example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

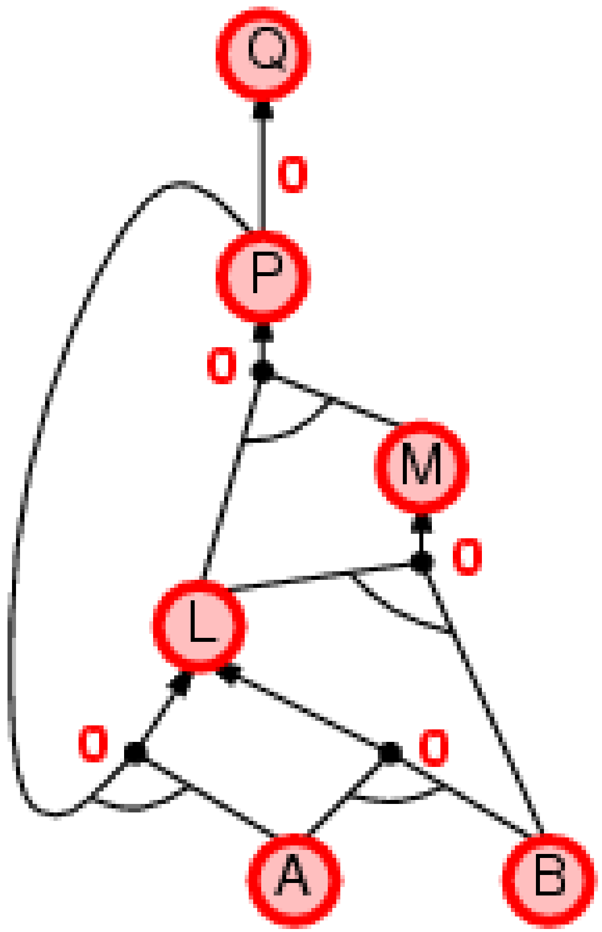
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining

- Assumption: KB of definite clauses
- Sound?
  - Every inference is an application of Modus Ponens.
- Complete?
  - Every entailed atomic sentence will be derived?
- Humans keep forward chaining under careful control (to avoid irrelevant consequences)

Proof of completeness

1. Consider the final state of the inferred table (fixed point where no new atomic sentences are derived)

2. Consider the final state as a model m, assigning $true$ to inferred symbols and $false$ to others

3. Every definite clause in the original *KB* is true in m

   If "$a_1 \wedge \cdots \wedge a_n \Rightarrow b$" is false then $a_1 \wedge \cdots \wedge a_n$ is true and b is false. But it contradicts with reaching a fixed point.

4. Thus, m is a model of *KB*

5. If $KB \vDash q$, atomic sentence q is true in every model of *KB*, including m

Backward chaining

- Idea: work backwards from the query q:
- To prove q:

  if q is known to be true then

  no work is needed

  else

  Find those rule concluding q

  If all premises of one of them can be proved (by backward chaining) then q is true

- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal has already been proved true or failed

Backward chaining example

$P \Rightarrow \boxed{Q}$

$L \wedge M \Rightarrow P$

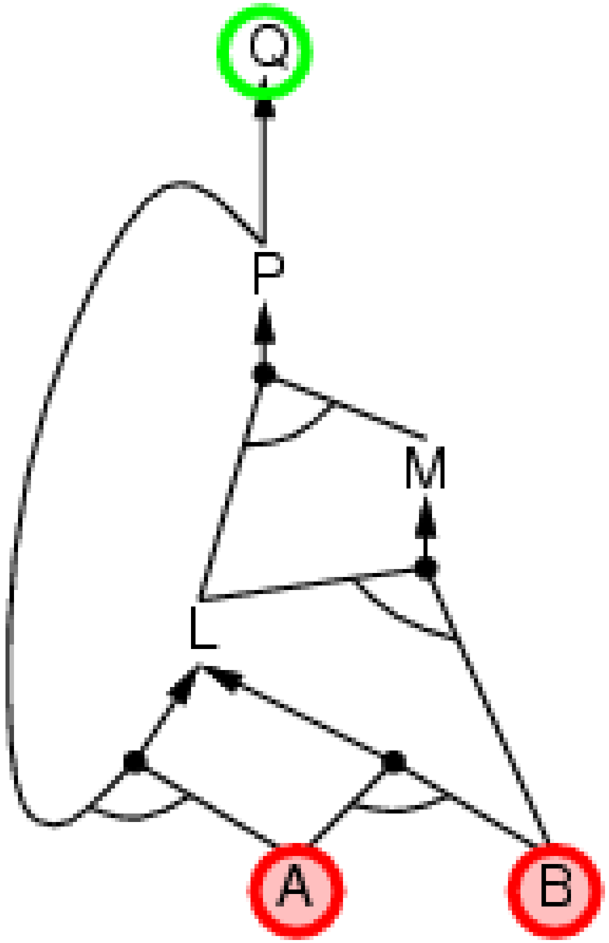
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\boxed{P} \Rightarrow \boxed{Q}$

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

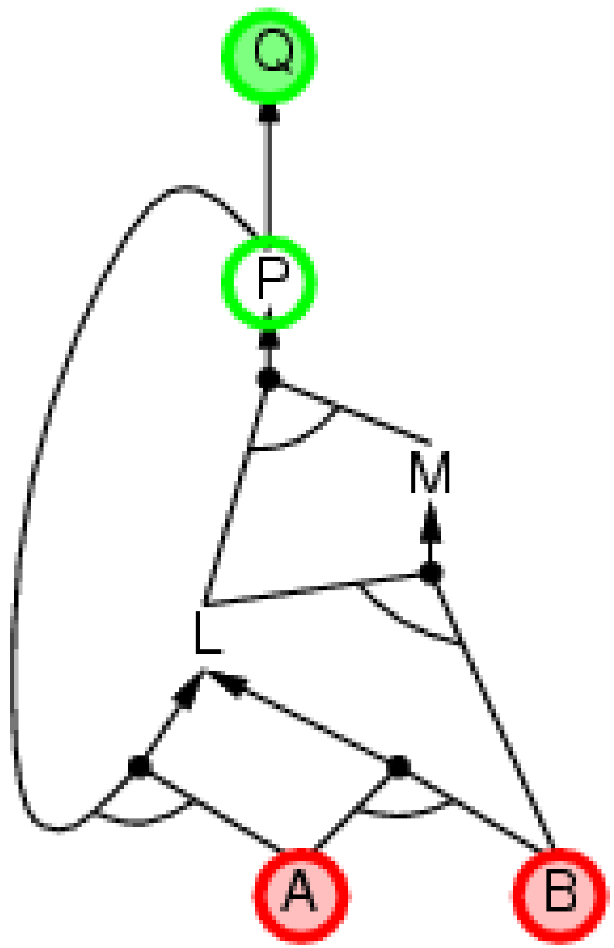
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$\boxed{P} \Rightarrow \boxed{Q}$

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

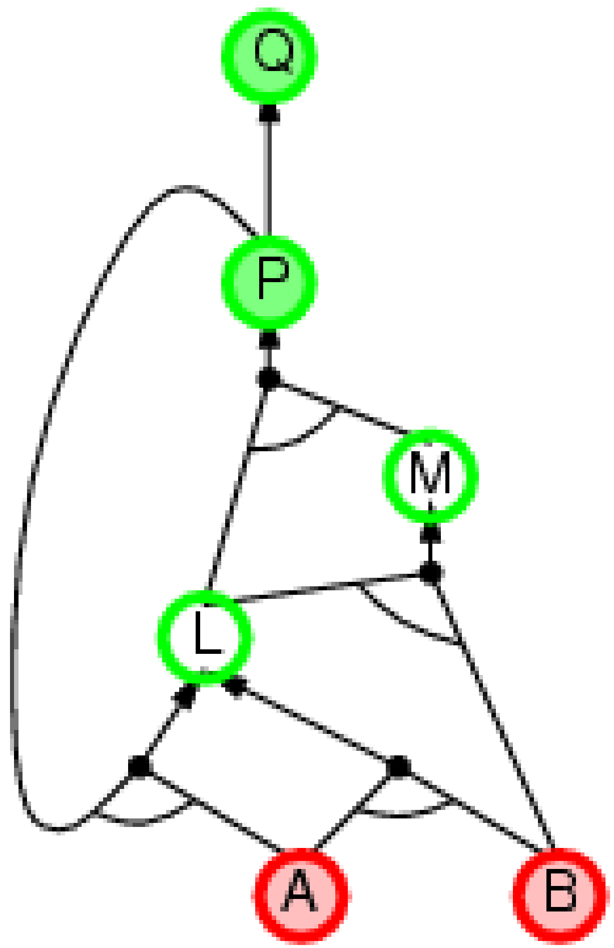
$B \wedge L \Rightarrow \boxed{M}$

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

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

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

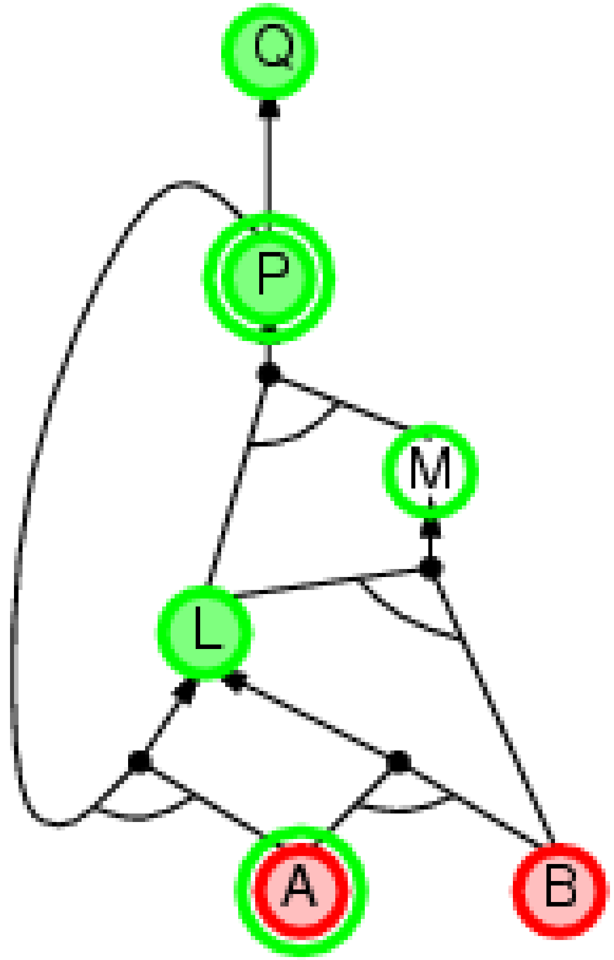
$B \wedge L \Rightarrow M$

$A \wedge \times \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

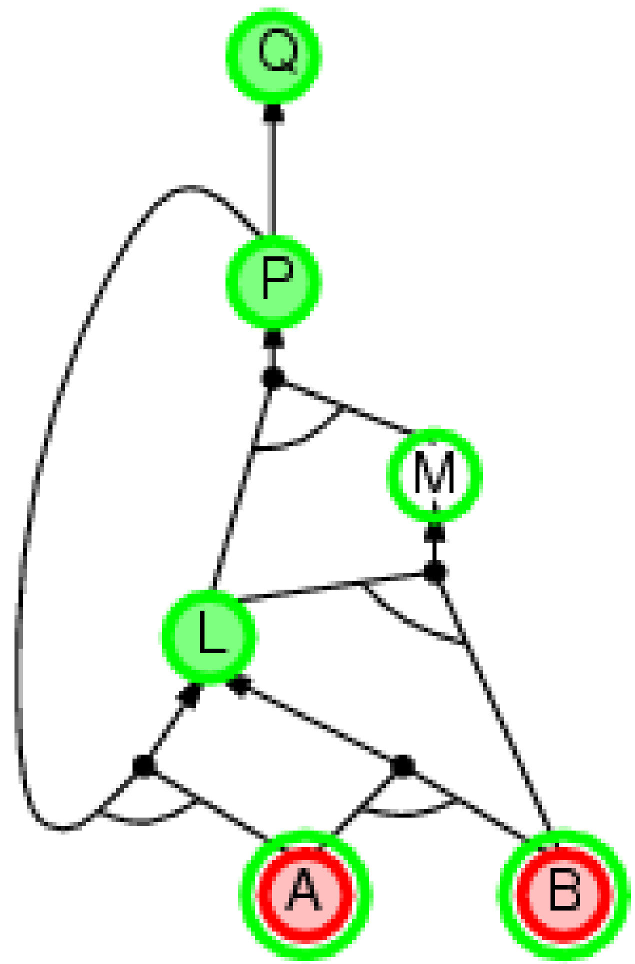
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

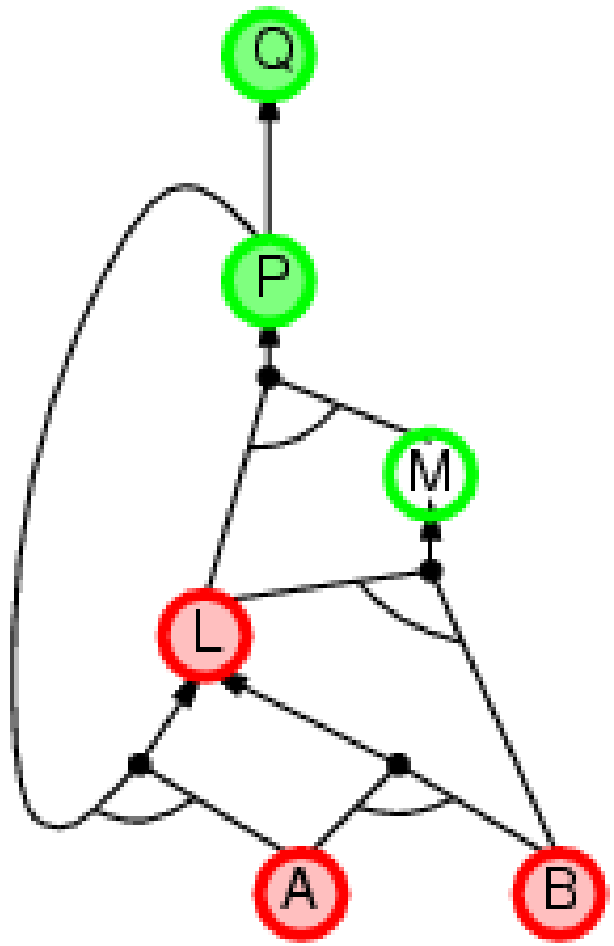
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

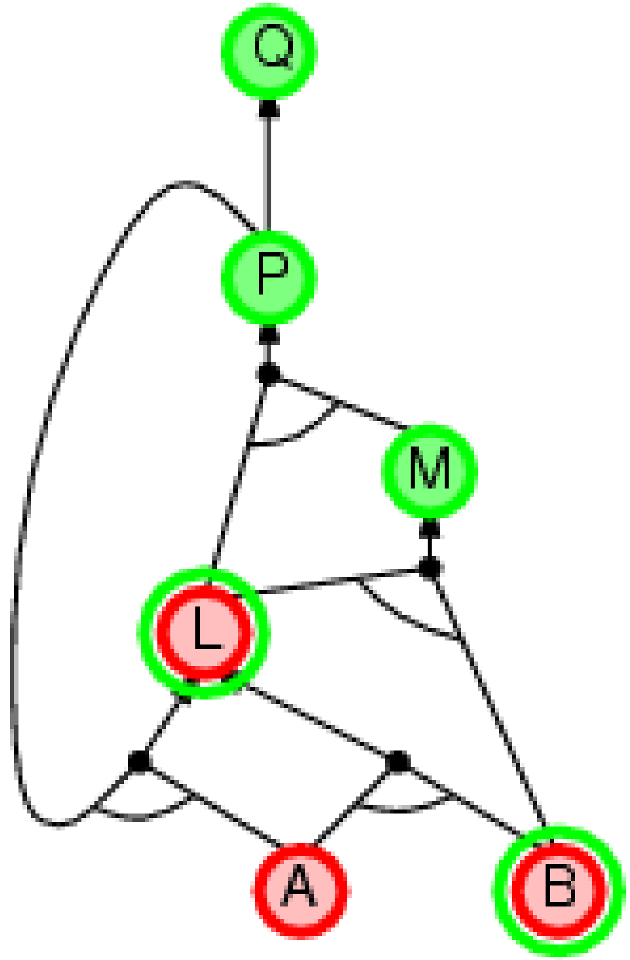
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

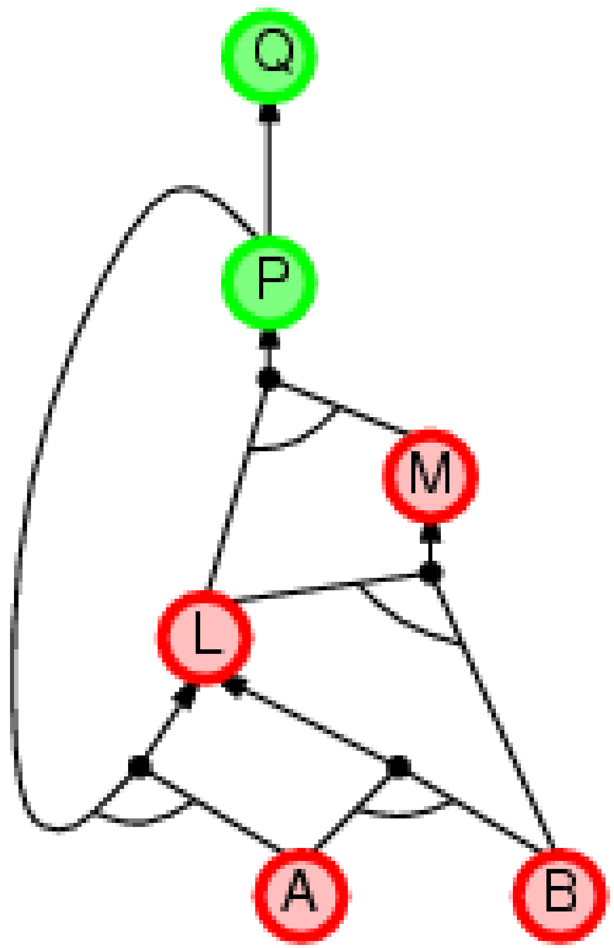
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

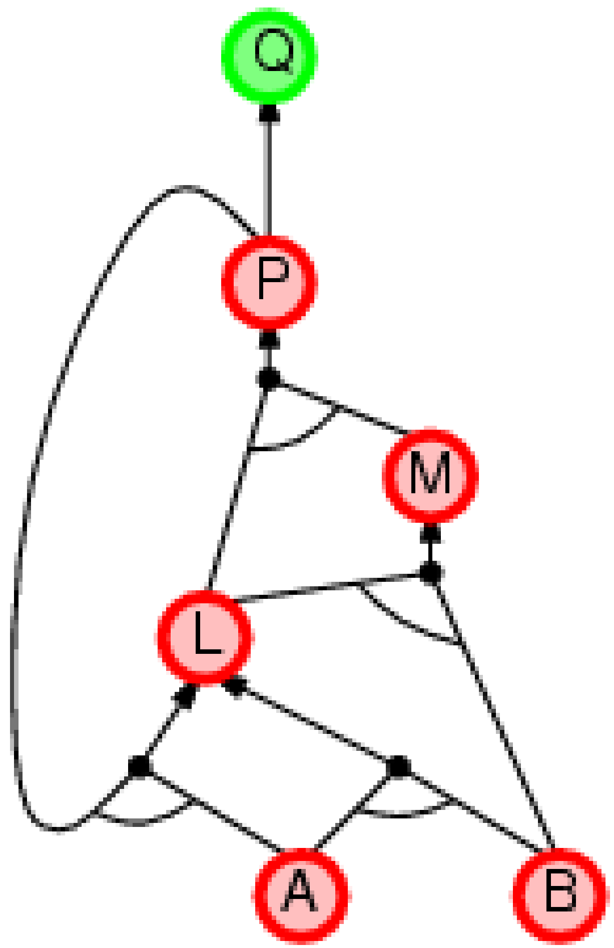
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

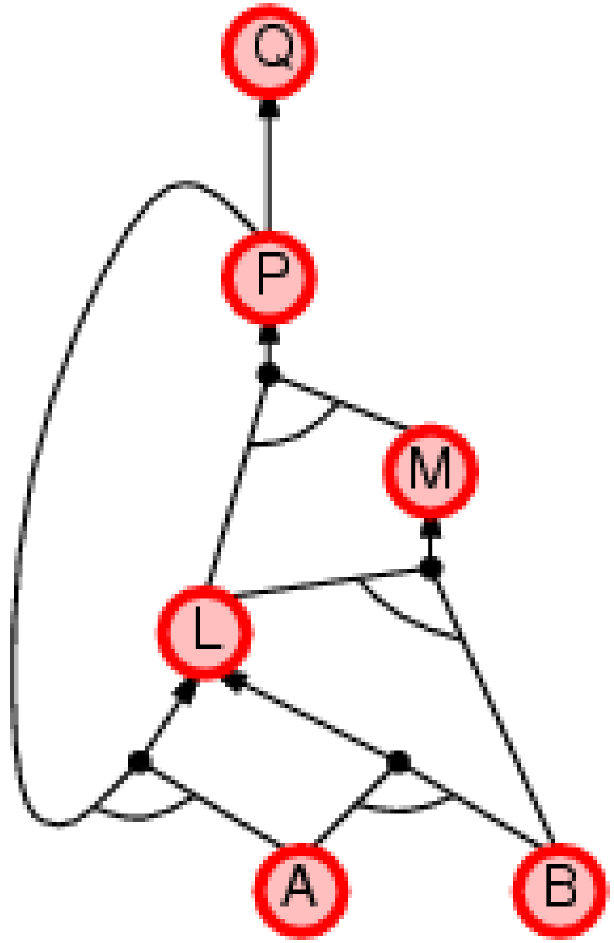
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward vs. backward chaining

- Forward chaining is data-driven, automatic, unconscious processing
  - May do lots of work that is irrelevant to the goal

- Backward chaining is goal-driven, appropriate for problem-solving
  - Only relevant facts are considered.

- Complexity of BC can be much less than linear in size of KB

Efficient propositional inference

- Efficient algorithms for general propositional inference:
  - We introduced DPLL algorithm as a systematic search method
    - Backtracking search algorithms
  - Now, we will introduce WalkSAT algorithm as a local search algorithm

WalkSAT

- It tests entailment $KB \vDash \alpha$ by testing unsatisfiability of $KB \wedge \neg\alpha$.

- Local search algorithms (hill climbing, SA, …)
  - state space: complete assignments
  - evaluation function: number of unsatisfied clauses

- WalkSAT
  - Simple & effective
  - Balance between greediness and randomness
    - To escape from local minima

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
```

If $\max\_flips = \infty$ and $p > 0$, WalkSAT will find a model (if any exists)
If $\max\_flips = \infty$, it never terminates for unsatisfiable sentences.

WalkSAT

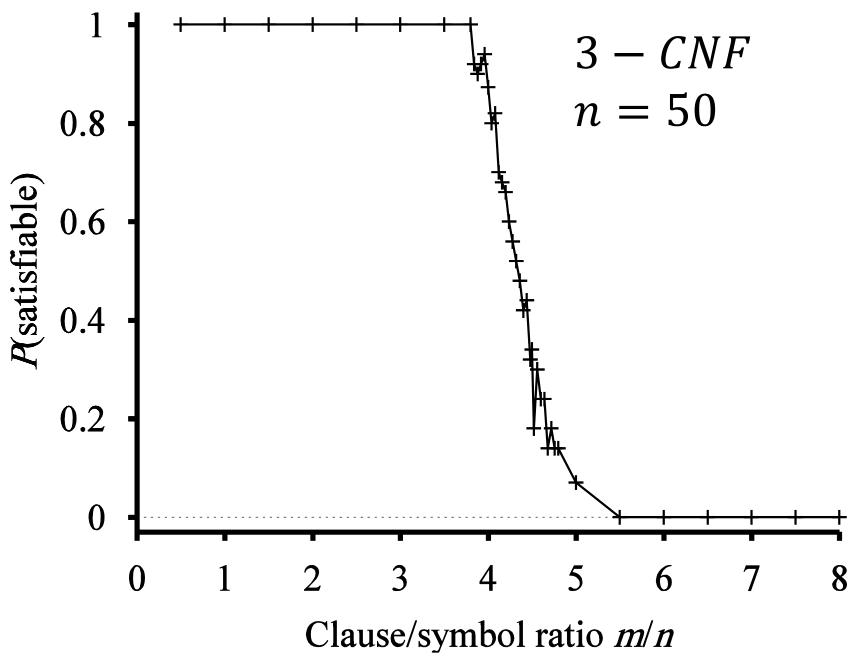
- Failure may mean:
  1) The sentence is unsatisfiable.
  2) The algorithm needs more time.
- It can not detect unsatisfiability.
  - Cannot be reliably used for deciding entailment.

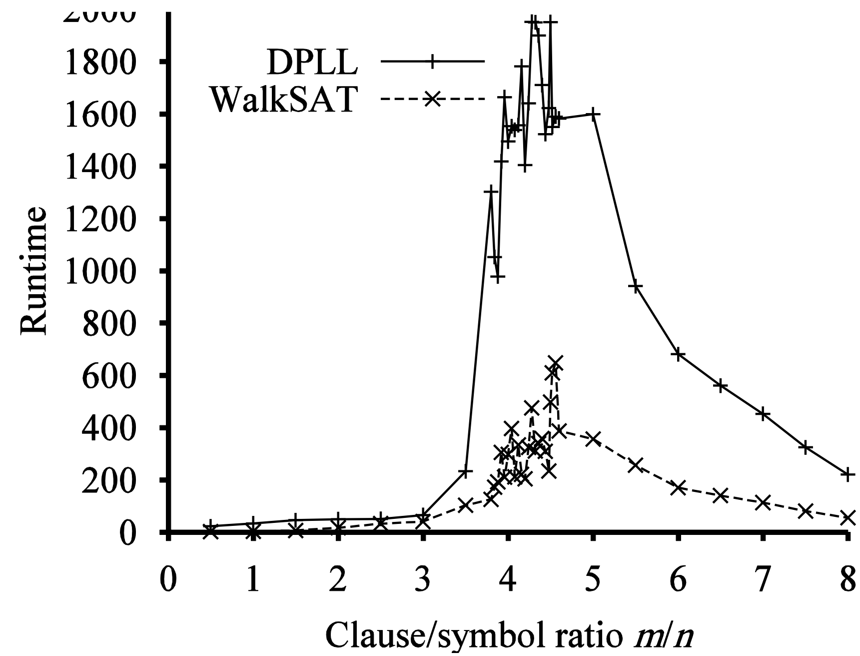
Random SAT problems

- Under-constrained problems: relatively few clauses
  - A large portion of possible assignments are solutions
    - e.g., $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$

- Over-constrained problems: many clauses relative to the number of variables
  - Likely to have no solution

Hard satisfiability problems





Hard problems around critical point

m = number of clauses
n = number of symbols

m/n = 4.3 (critical point)

Wumpus world example: propositional symbols

- Atemporal variables that may be needed
  - $P_{i,j}$ is true if there is a pit in $[i, j]$.
  - $W_{i,j}$ is true if there is a wumpus in $[i, j]$, dead or alive.
  - $B_{i,j}$ is true if the agent perceives a breeze in $[i, j]$.
  - $S_{i,j}$ is true if the agent perceives a stench in $[i, j]$.

- Some of the temporal variables that may be needed
  - Variables for percepts and actions
  - $L_{i,j}^{t}, FacingEast^{t}, FacingWest^{t}, FacingNorth^{t}, FacingSouth^{t}$
  - $HaveArrow^{t}$
  - $WumpusAlive^{t}$

Wumpus world example: axioms on atemporal aspect of the world

- Axioms: general knowledge about how the world works
- General rules on atemporal variables

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

$$\neg W_{1,1}$$

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

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

A distinct rule is needed for each square

$$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$$

$$\neg W_{1,1} \vee \neg W_{1,2}$$

$$\neg W_{1,1} \vee \neg W_{1,3}$$

…

Wumpus world example: perceptions & actions

- Perceptions are converted to facts and are Telled to KB
  - $MAKE\_PERCEPT\_SENTENCE([Breeze, Stench, None, None, None], t)$
  - $Breeze^t, Stench^t$ are added to KB

- Selected action is converted to a fact and is Telled to KB
  - $MAKE\_ACTION\_SENTENCE(Forward, t)$
  - $Forward^t$ is added to KB

Wumpus world example: transition model

- Changing aspect of world (Temporal variables)
  - They are also called as fluents or state variables

- Initial KB includes the initial status of some of the temporal variables:
  - $L_{1,1}^{0}, FacingEast^{0}, HaveArrow^{0}, WumpusAlive^{0}$

- Percepts are connected to fluents describing the properties of squares
  - $L_{x,y}^{t} \Rightarrow (Breeze^{t} \Leftrightarrow B_{x,y})$
  - $L_{x,y}^{t} \Rightarrow (Stench^{t} \Leftrightarrow S_{x,y})$
  - …

- Transition model: fluents can change as the results of agent's actions

Wumpus world example: transition model
Frame problem

- Transition model:

  $L_{1,1}^{0} \wedge FacingEast^{0} \wedge Forward^{0} \Rightarrow (L_{2,1}^{1} \wedge \neg L_{1,1}^{1})$

  $L_{1,1}^{0} \wedge FacingEast^{0} \wedge TurnLeft^{0} \Rightarrow (L_{1,1}^{1} \wedge FacingNorth^{1} \wedge \neg FacingEast^{1})$

  $\ldots$

- After doing $Forward$ action at time 0:
  - $ASK(KB, HaveArrow^{1}) = false$
    - Why?

- Frame problem: representing the effects of actions without having to represent explicitly a large number of intuitively obvious non-effects

Wumpus world example
Solution to frame problem: successor-state axioms

- A Solution to frame problem: Successor-state axioms

  Instead of writing axioms about actions, we write axioms about fluents

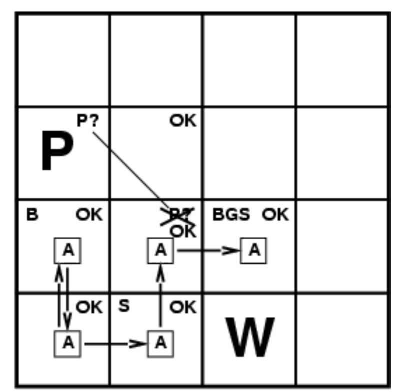
  - $F^{t+1} \Leftrightarrow ActionCausesF^{t} \vee (F^{t} \wedge \neg ActionCausesNotF^{t})$

$$HaveArrow^{t+1} \Leftrightarrow (HaveArrow^{t} \wedge \neg Shoot^{t})$$

$$\begin{aligned} L_{1,1}^{t+1} \Leftrightarrow\ & (L_{1,1}^{t} \wedge (\neg Forward^{t} \vee Bump^{t+1})) \\ & \vee (L_{1,2}^{t} \wedge (FacingSouth^{t} \wedge Forward^{t})) \\ & \vee (L_{2,1}^{t} \wedge (FacingWest^{t} \wedge Forward^{t})) \end{aligned}$$

A hybrid agent for the wumpus world

- Basic concepts:
  - Safeness of $[x, y]$ square:
    - $OK_{x,y}^{t} \Leftrightarrow \neg P_{x,y} \wedge \neg (W_{x,y} \wedge \ WumpusAlive^{t})$
  - Set of safe squares:
    - $safe \leftarrow \{[x,y]: ASK(KB, OK_{x,y}^{t}) = true\}$
  - Set of unvisited squares:
    - $unvisited \leftarrow \{[x,y]: ASK(KB, L_{x,y}^{t'}) = false \ for \ all \ t' \leq t\}$



```
function HYBRID_WUMPUS_AGENT(percept) returns an action sequence
  inputs: percept, a list, [breeze, stench, glitter, bump, scream]
  persistent: KB, initially the atemporal "wumpus physics"
              t, a time counter, initially 0
              plan, an action sequence, initially empty

  TELL(KB, MAKE_PERCEPT_SENTENCE(percept, t))
  TELL the KB the temporal "physics" sentences for time t
  safe ← {[x, y]: ASK(KB, OK^t_{x,y}) = true}
  if ASK(KB, Glitter^t) = true then
    plan ← [Grab] + PLAN_ROUTE(current, {[1,1]}, safe) + [Climb]
  if plan = {} then
    unvisited ← {[x, y]: ASK(KB, L^{t'}_{x,y}) = false for all t' ≤ t}
    plan ← PLAN_ROUTE(current, unvisited ∩ safe, safe)
  if plan = {} and ASK(KB, HaveArrow^t) = true then
    possible_wumpus ← {[x, y]: ASK(KB, ¬W_{x,y}) = false}
    plan ← PLAN_SHOT(current, possible_wumpus, safe)
  if plan = {} then
    not_unsafe ← {[x, y]: ASK(KB, ¬OK^t_{x,y}) = false}
    plan ← PLAN_ROUTE(current, unvisited ∩ not_unsafe, safe)
  if plan = {} then
    plan ← PLAN_ROUTE(current, {[1,1]}, safe) + [Climb]
  action ← POP(plan)
  TELL(KB, MAKE_ACTION_SENTENCE(action, t))
  t ← t + 1
  return action
```

Hybrid agent (phases listed based on priority)

- Simple-reflex (condition-action rule):

if $ASK(KB, Glitter^t) = true$ **then**

$plan \leftarrow [Grab] + PLAN\_ROUTE(current, \{[1,1]\}, safe) + [Climb]$

- Goal-based & reasoning:

if $plan = \{\}$ **then**

$unvisited \leftarrow \{[x,y]: ASK(KB, L_{x,y}^{t'}) = false\ for\ all\ t' \leq t\}$

$plan \leftarrow PLAN\_ROUTE(current, unvisited \cap safe, safe)$

if $plan = \{\}$ and $ASK(KB, HaveArrow^t) = true$ **then**

$possible\_wumpus \leftarrow \{[x,y]: ASK(KB, \neg W_{x,y}) = false\}$

$plan \leftarrow PLAN\_SHOT(current, possible\_wumpus, safe)$

Hybrid agent (phases listed based on priority)

- Exploration & reasoning

if $plan = \{\}$ **then**

$not\_unsafe \leftarrow \{[x,y]: ASK\left(KB, \neg OK_{x,y}^{t}\right) = false\}$

$plan \leftarrow PLAN\_ROUTE(current, unvisited \cap not\_unsafe, safe)$

- Goal-based:

if $plan = \{\}$ **then**

$plan \leftarrow PLAN\_ROUTE(current, \{[1,1]\}, safe) + [Climb]$

Hybrid agent: Using A* to make a plan

function $PLAN\_ROUTE(current, goal, allowed)$ **returns** an action sequence
  inputs: $current$, the agent's current position
    $goals$, a set of squares; try to plan a route to one of them
    $allowed$, a set of squares that can form part of the route

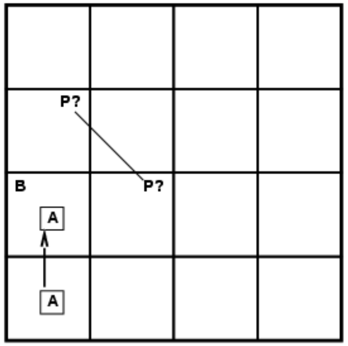
  $problem \leftarrow ROUTE\_PROBLEM(current, goals, allowed)$
  return $A^{*}\_GRAPH\_SEARCH(problem)$

Logical state estimation

- Problem: Computational time of ASK grows by the length of the agent's life
  - We must save the results of inference to use them in the next time steps

- Solution: Belief state instead of the whole past history of percepts and actions
  - Belief state: the agent's knowledge about the state of the world (the set of all possible current states of the world)

$$B_{1,2} \wedge (P_{1,3} \vee P_{2,2})$$
$$\wedge\, L_{1,2}^{1} \wedge FacingNorth^{1} \wedge HaveArrow^{1} \wedge WumpusAlive^{1}$$



  - State estimation: process of updating the belief state when doing actions and receiving new percepts

Approximate state estimation

- Problem: The exact belief state estimation may require logical formulas whose size is exponential in the number of symbols.
  - n fluent symbols, 2^n possible physical states, 2^{2^n} possible belief states

- Solution: approximation
  - e.g., 1-CNF belief state: given the belief state at $t-1$, agent tries to prove X^t or $\neg X^t$ for each X^t)

- Approximate belief state estimation?

compact representation (open area for research)

- Relation to model-based agents

Limitations of propositional logic

- Wumpus world
  - Distinct rules "for each square [x,y]" and "for each time t"
  - $L_{x,y}^{t} \wedge FacingEast^{t} \wedge TurnLeft^{t} \Rightarrow (L_{x,y}^{t+1} \wedge FacingUp^{t+1} \wedge \neg FacingEast^{t+1})$
- Many rules are needed to describe the world axioms
  - Rather impractical for bigger world
- We can not express general knowledge about "physics" of the world directly in propositional language
  - More expressive language is needed.
- Qualification problem: specifying all the exceptions
  - Probability theory allows us to summarize all the exceptions (without explicitly naming them)