CS3243: Introduction to Artificial Intelligence

Semester 2, 2020

Logical Agents

AIMA Chapter 7

Knowledge-Based Agents

- Until now – trying to find an optimal solution via **search**.
- No real model of what the agent **knows**.
- This class: **represent** agent domain **knowledge** using **logical formulas.**

Knowledge Base (KB)

Inference Engine ← Domain-independent algorithms

Knowledge Base ← Domain-specific content

- Knowledge base = set of sentences in a formal language
- Declarative approach to building an agent (or other system):
  - TELL it what it needs to know
- Then it can ASK itself what to do - answers should follow from the KB
- Agents can be viewed at the knowledge level

  i.e., specify knowledge and goals, regardless of implementation
- Or at the implementation level
  - i.e., data structures in KB and algorithms that manipulate them

What is the best action at time t?

What did I perceive at time t?

What happened?

What have I done?

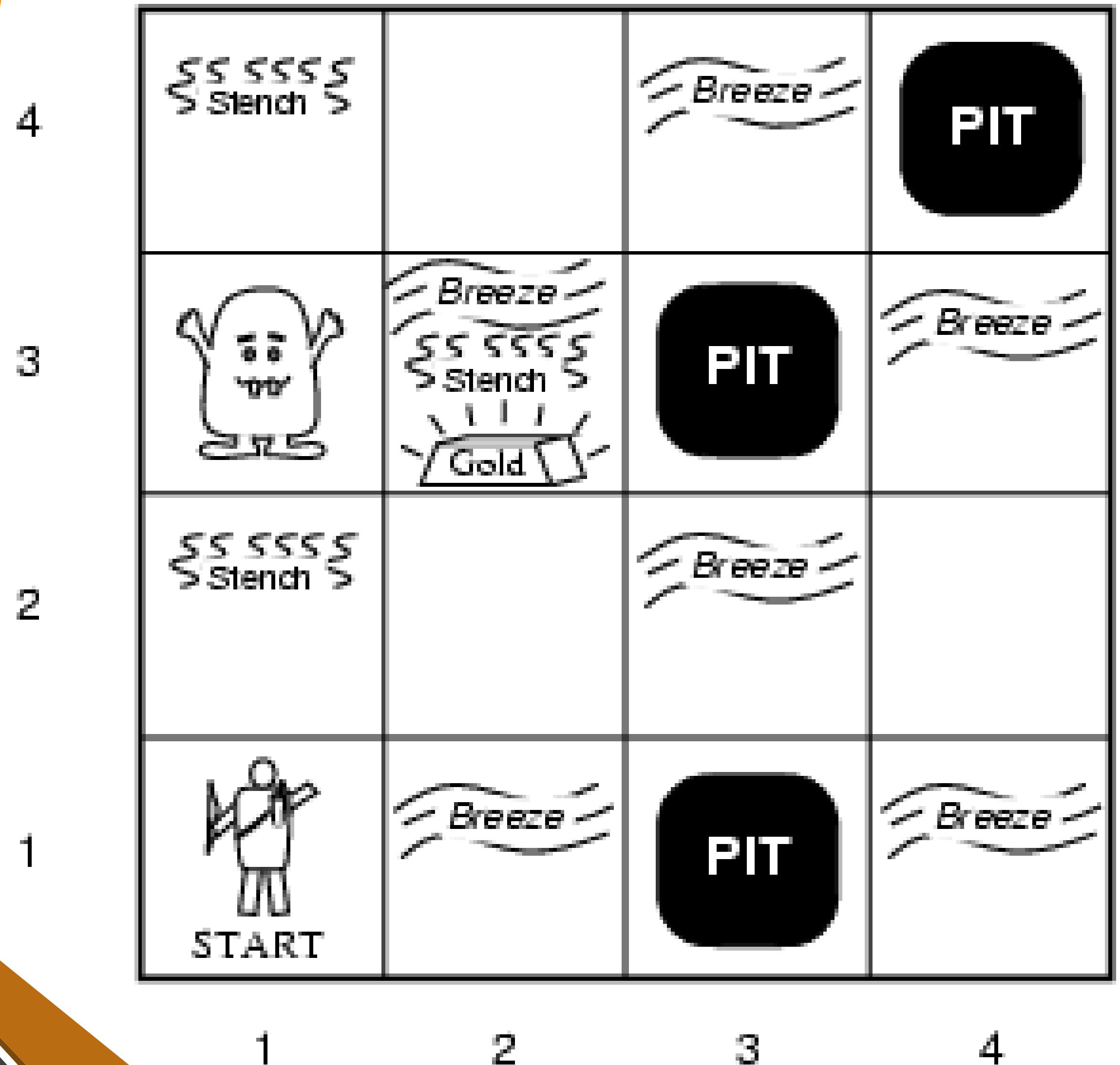
```
function KB-A   T( percep      urns an action
   persistent: K    knowled   base
               t, a c   nter, in   ally 0, indicating time

   TELL(KB, MAKE-P  RCEPT-SENTENCE(percept, t))
   action ← ASK(KB, MAKE-ACTION-QUERY(t))
   TELL(KB, MAKE-ACTION-SENTENCE(action, t))
   t ← t + 1
   return action
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal world representations
- Deduce hidden world properties, and deduce actions

Wumpus World



Performance Measure?

Environment?

Actuators?

Sensors?

Performance measure

- gold $+1000$, death -1000
- -1 per action, -10 for using the arrow

Environment

- 4×4 grid of rooms
- agent, wumpus, gold, pits

Actuators

- Turn left/right, Forward
- Shoot: kills wumpus if facing it; uses up the only arrow
- Grab: picks up gold if in same square
- Climb: get out of cave if in [1,1]

Sensors

- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Gets bumped if agent walks into a wall
- Hears scream if wumpus killed

Properties of Wumpus World

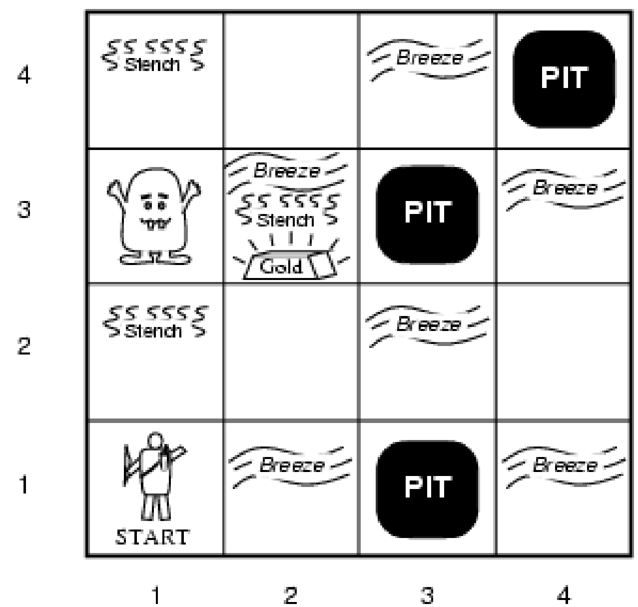
Fully Observable?	No – only local perception
Deterministic?	Yes
Episodic?	No – sequential actions
Static?	Yes – nothing moves
Discrete?	Yes
Single-Agent?	Yes

Exploring a Wumpus World

Agent’s view

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **P?**	3,2	4,2
1,1 **V** $\boxed{\mathbf{A}}$ **OK**	2,1 **B** $\boxed{\mathbf{A}}$ **OK**	3,1 **P?**	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe Square
P = Pit
S = Stench
V = Visited
W = Wumpus



Exploring a Wumpus World

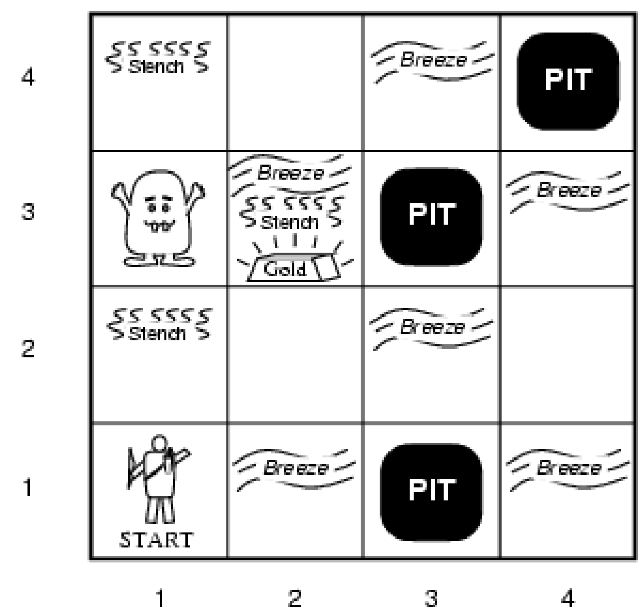
Agent's view

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 S A OK	2,2 P? OK	3,2	4,2
1,1 V A OK	2,1 B A OK	3,1 P?	4,1

No Breeze!

No Stench at [2,1]

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe Square
P = Pit
S = Stench
V = Visited
W = Wumpus

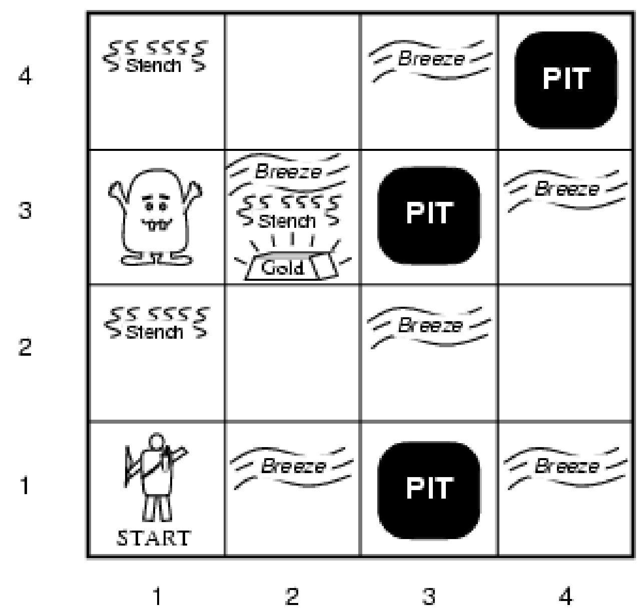


Exploring a Wumpus World

Agent's view

1,4	2,4	3,4	4,4
1,3 **W!**	2,3 $\boxed{A}$ **OK**	3,3	4,3
1,2 **S** $\boxed{V}$ **OK**	2,2 $\boxed{V}$ **OK**	3,2 **OK**	4,2
1,1 **V** **OK**	2,1 **B V** **OK**	3,1 **P!**	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe Square
P = Pit
S = Stench
V = Visited
W = Wumpus

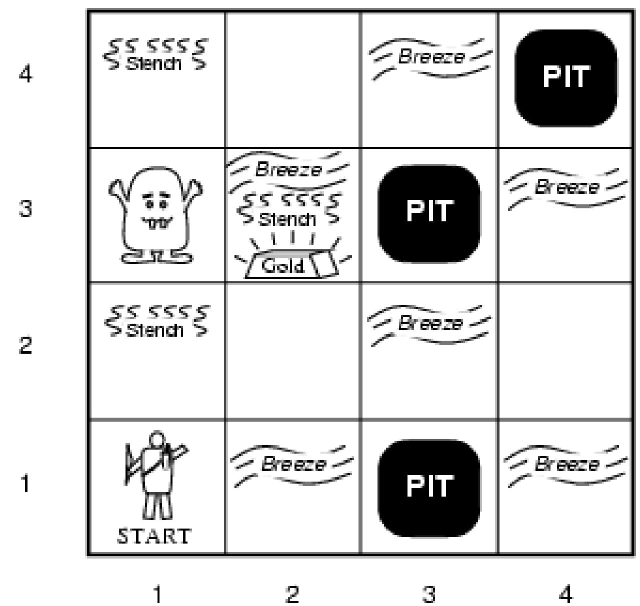


Exploring a Wumpus World

Agent's view

1,4	2,4	3,4	4,4
1,3 **W!**	2,3 **A** **S B G** **OK**	3,3	4,3
1,2 **S V** **OK**	2,2 **V** **OK**	3,2 **OK**	4,2
1,1 **V** **OK**	2,1 **B V** **OK**	3,1 **P!**	4,1

A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe Square
P = Pit
S = Stench
V = Visited
W = Wumpus



Logic in General

- Logic: formal language for KR, infer conclusions
- Syntax: defines the sentences in the language
- Semantics: define the "meaning" of sentences;
  - i.e., define truth of a sentence in a world
- E.g., language of arithmetic
  - $x + 2 \geq y$ is a sentence; $x2y +>$ is not a sentence
  - $x + 2 \geq y$ is true in a world where $x = 7, y = 1$
  - $x + 2 \geq y$ is false in a world where $x = 0, y = 6$

Entailment

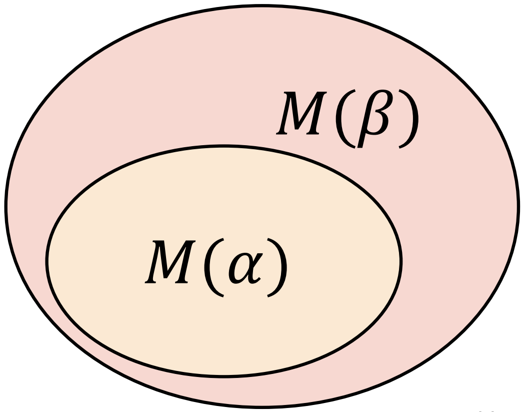
- Modeling: m models α if α is true under m.
  For example, what are models for the following?

$$\alpha = (q \in \mathbb{Z}_+) \wedge (\forall n, m \in \mathbb{Z}_+ : q = nm \Rightarrow n \vee m = 1)$$

- We let $M(\alpha)$ be the set of all models for α
- Entailment means that one thing follows from another:

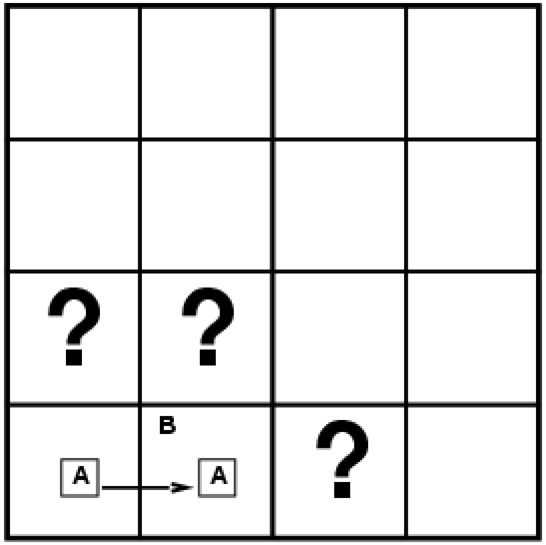
$$\alpha \vDash \beta \text{ or equivalently } M(\alpha) \subseteq M(\beta)$$

- For example:
  $\alpha = (q \text{ is prime})$ entails
  $\beta = (q \text{ is odd}) \vee (q = 2)$.

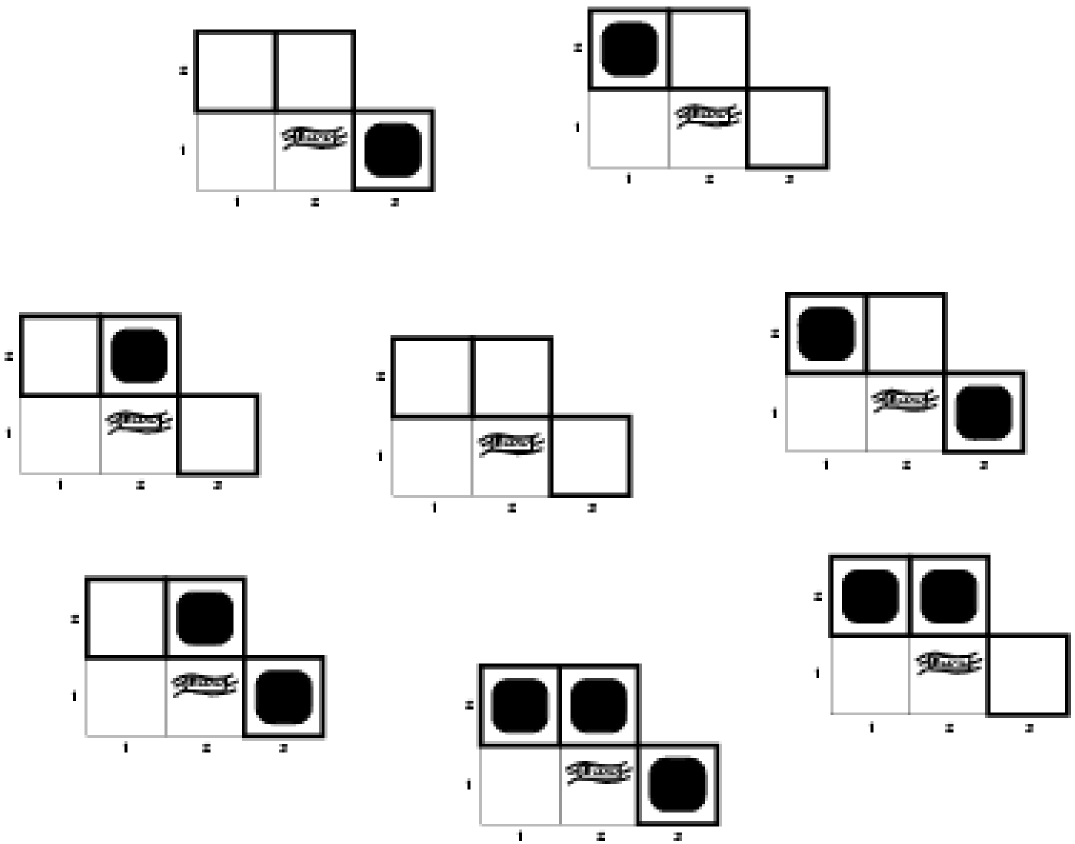


Entailment in the Wumpus World

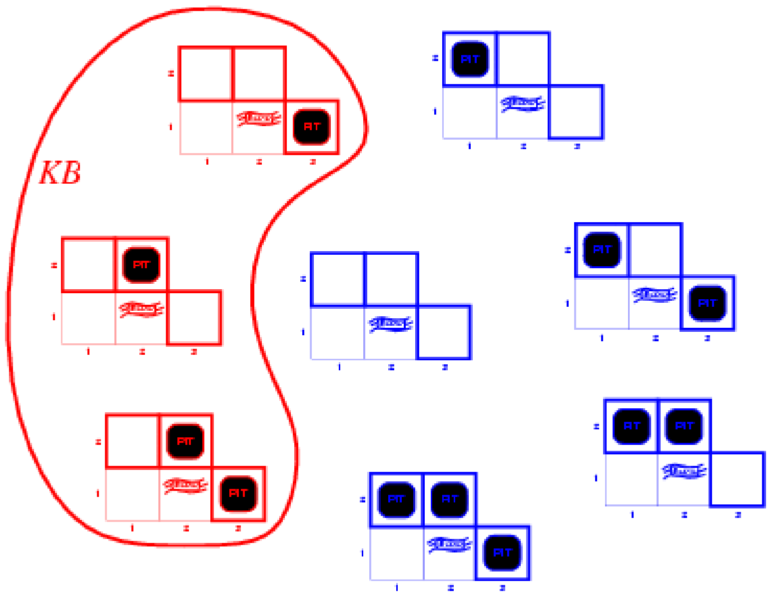
- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for KB assuming only pits
- 3 Boolean choices $\Rightarrow$ 8 possible models



Wumpus Models

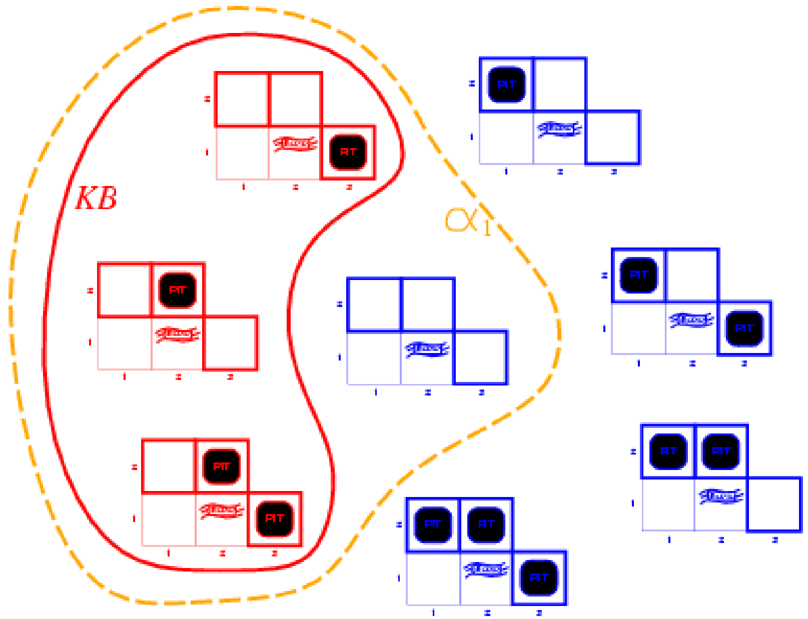


Wumpus Models



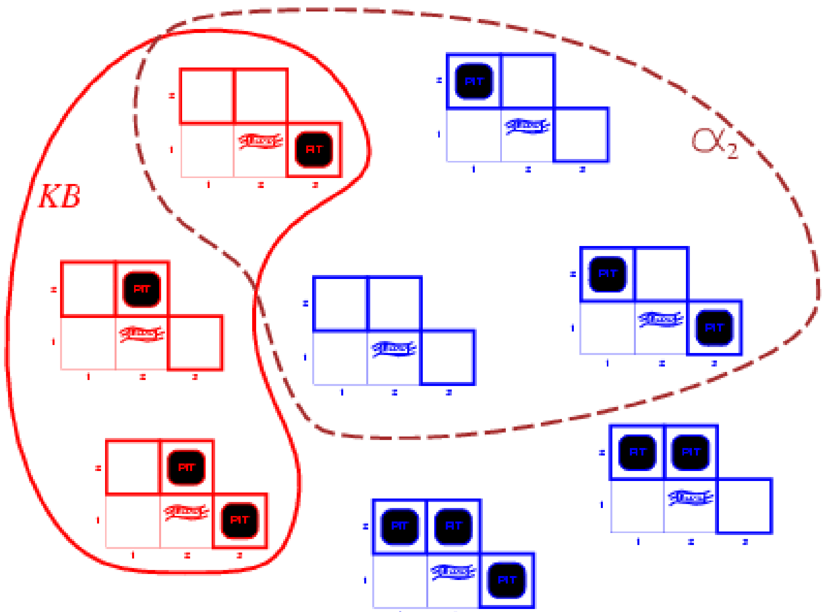
- KB = wumpus-world rules + percepts

Wumpus Models



- KB = wumpus-world rules + percepts
- α_1 = “[1,2] is safe”, $KB \vDash \alpha_1$, proved by model checking
- The agent can infer that [1,2] is safe

Wumpus Models



- KB = wumpus-world rules + percepts
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$
- The agent cannot infer that [2,2] is safe (or unsafe)!

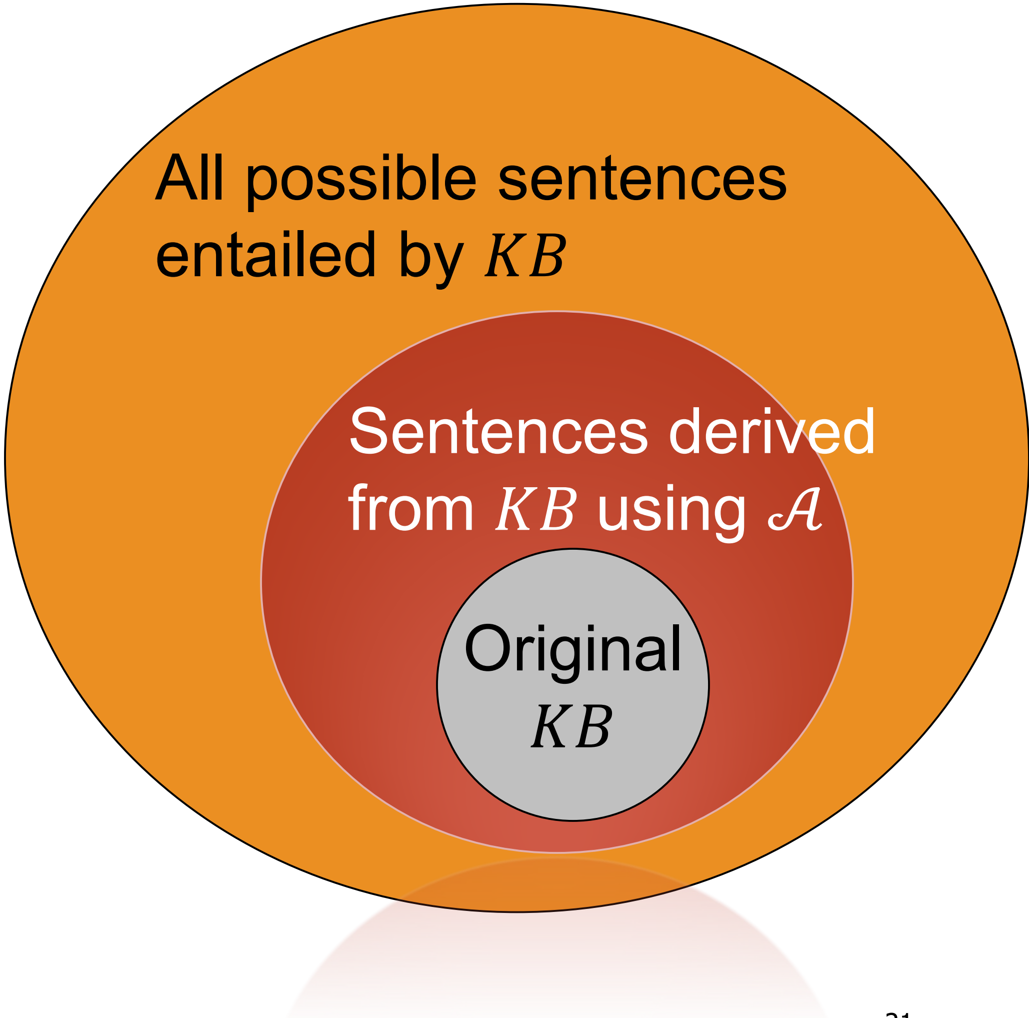
Inference algorithm: is a sentence α is derived from KB?

- Define $KB \vdash_{\mathcal{A}} \alpha$ to be "sentence α is derived from KB by inference algorithm $\mathcal{A}$"
  - $\mathcal{A}$ is **sound** if $KB \vdash_{\mathcal{A}} \alpha$ implies $KB \vDash \alpha$.
    "don't infer nonsense"
  - $\mathcal{A}$ is **complete** if $KB \vDash \alpha$, implies $KB \vdash_{\mathcal{A}} \alpha$.
    "If it's implied, it can be inferred"

Is an inference algorithm complete and sound?

Completeness: $\mathcal{A}$ **is complete if whenever** $KB \vDash \alpha$**, it is also true that** $KB \vdash_{\mathcal{A}} \alpha$

- An incomplete inference algorithm cannot reach all possible conclusions
- Equivalent to completeness in search (chapter 3)



Propositional Logic: Syntax

- A simple logic – illustrates basic ideas
- Defines allowable sentences
- Sentences are represented by symbols e.g. S_1,S_2
- Logical connectives for constructing complex sentences from simpler ones:
  - 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)
    - $S_1 \vee S_2$ is a sentence (disjunction)
    - $S_1 \Rightarrow S_2$ is a sentence (implication)
    - $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional Logic: Semantics

A model is then just a **truth assignment to the basic variables**.

If a model has n variables, how many truth assignments are there?

All other sentences' truth value is derived according to logical rules.

$$x_1 = T; x_2 = F; x_3 = T$$

$$(x_1 \land \neg x_2) \Rightarrow \neg\big(x_3 \lor (\neg x_1 \land x_2)\big) = ?$$

Knowledge Base for Wumpus World

- $P_{ij} = \text{True} \Leftrightarrow$ there is a pit in $[i, j]$.
- $B_{ij} = \text{True} \Leftrightarrow$ there is breeze in $[i, j]$
- Rules:
  - $R_1 : \neg P_{1,1}$
  - $R_4 : \neg B_{1,1}$
  - $R_5 : P_{2,1}$

KB is true iff $\wedge_{k=1,\ldots,5} R_k$ is true

- "Pits cause breezes in adjacent squares"
  - $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Inference

- Given a knowledge base, infer something non-obvious about the world.
- Mimic logical human reasoning
- After exploring 3 squares, we have some understanding of the Wumpus world
- Inference ⇨ Deriving knowledge out of percepts

Given KB and α, we want to know if $KB \vdash \alpha$

Truth Table for Inference

Is α_1 true whenever KB is true?

		$_{,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
		se	$false$	$false$	$false$	$false$	$false$	$true$
$alse$	$false$	$false$	$false$	$false$	$false$	$true$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$false$	$true$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$\underline{true}$	$\underline{true}$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$false$	$true$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$	$false$

$R_1: \neg P_{1,1}$

$R_4: \neg B_{1,1}$

$R_5: B_{2,1}$

$$\alpha_1 = \neg P_{1,2}$$

Does KB entail α_1?

Can we infer that [1,2] is safe from pits?

Inference by Truth-Table Enumeration

- Depth-first enumeration of all models is sound and complete
- For n symbols, time complexity is $\mathcal{O}(2^n)$, space complexity is $\mathcal{O}(n)$

```
function TT-ENTAILS?(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, { })
─────────────────────────────────────────────────────────────
function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true // when KB is false, always return true
   else do
      P ← FIRST(symbols)
      rest ← REST(symbols)
      return (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
             and
             TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

Check all possible truth assignments

Validity and Satisfiability

A sentence is valid if it is true in all models,

e.g., $True$, $A\square\ A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to entailment via the Deduction Theorem:

$KB \vDash \alpha$ iff $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in no models

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

Satisfiability is connected to entailment via the following:
$KB \vDash \alpha$ **if and only if** $(KB \wedge \neg\alpha)$ **is unsatisfiable**

Proof Methods

Applying inference rules (aka theorem proving)

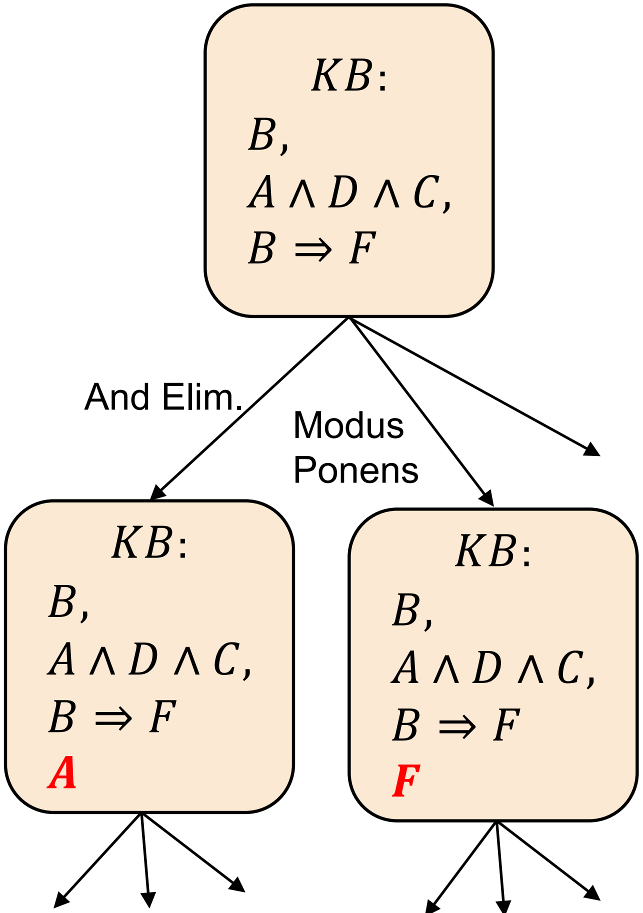
- Generation of new sentences from old
- Proof = sequential application of inference rules
- Inference rules are actions in a search algorithm
- Proof can be more efficient: ignores irrelevant propositions
- Transformation of sentences into a normal form

Model checking

- Truth table enumeration (time complexity exponential in n)
- Improved backtracking, e.g. DPLL
- local search in model space (sound but incomplete), e.g. min-conflicts-like hill-climbing algorithm

Applying Inference Rules

- Equivalent to a search problem
  - States: KBs (initial state is initial KB)
  - Actions: Inference rules
  - Transition: add sentence to current KB
  - Goal: KB contains sentence to prove
- Examples of inference rules
  - And-Elimination (A.E.): $a \wedge b \vDash a$
  - Modus Ponens (M.P.): $a \wedge (a \Rightarrow b) \vDash b$
  - Logical Equivalences: $(a \vee b) \vDash \neg(\neg a \wedge \neg b)$



Resolution for Conjunctive Normal Form (CNF)

- conjunction of "disjunctions of literals" (clauses)
- E.g., $(x_1 \lor \neg x_2) \land (x_2 \lor x_3 \lor \neg x_4)$
- Resolution: if a literal x appears in C_1 and its negation $\neg x$ appears in C_2, it can be deleted:

$$\frac{(x_1 \lor \cdots \lor x_m \lor x) \land (y_1 \lor \cdots \lor y_k \lor \neg x)}{(x_1 \lor \cdots \lor x_m \lor y_1 \lor \cdots \lor y_k)}$$

(delete duplicate variables as necessary)

- Resolution is **sound** and **complete** for propositional logic

Conversion to CNF: the Rules

1. Convert $\alpha \Leftrightarrow \beta$ to $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Convert $\alpha \Rightarrow \beta$ to $\neg\alpha \vee \beta$
3. Move $\neg$ inwards using De Morgan and double negation
    1. Convert $\neg(\alpha \vee \beta)$ to $\neg\alpha \wedge \neg\beta$
    2. Convert $\neg(\alpha \wedge \beta)$ to $\neg\alpha \vee \neg\beta$
    3. Convert $\neg(\neg\alpha)$ to α
4. Convert $(\alpha \vee (\beta \wedge \gamma))$ to $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Resolution Algorithm

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

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

What does an empty clause imply??

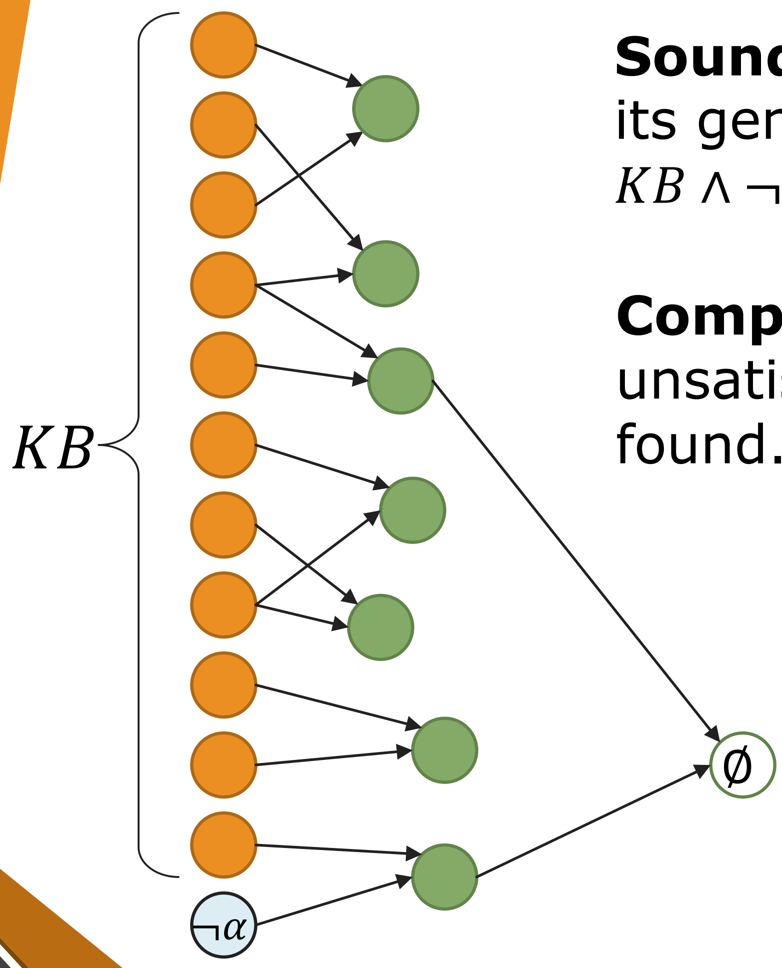
Proof by contradiction: show that $KB \wedge \neg\alpha$ is unsatisfiable

The resolution algorithm summary: repeatedly resolve two clauses from the list of clauses, and add the the result to list of clauses.

Keep on doing this until the empty clause is found.

Why is Resolution for CNF Sound and Complete?

Why is Resolution for CNF Sound and Complete?



Soundness: each resolvent is implied by its generating clauses. If $\emptyset$ is found, then $KB \wedge \neg\alpha$ was unsatisfiable…

Completeness: If $KB \wedge \neg\alpha$ is unsatisfiable, then $\emptyset$ will eventually be found.

Resolution Example

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

Prove that $\boldsymbol{KB \rightarrow \alpha}$ via proof by contradiction, i.e. assume that $\boldsymbol{KB \wedge \neg\alpha}$ holds and resolve to contradiction.

Forward and Backward Chaining

- Horn Form (restricted)
  - KB = conjunction of Horn clauses
  - Horn clause = definite clause or goal clause
    - Definite clause : $\bigwedge_j \alpha_j \Rightarrow \beta$
    - Goal clause : $\bigwedge_j \alpha_j \Rightarrow False$
  - e.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$
- Inference with Horn clauses: forward chaining or backward chaining algorithms. Easy to interpret, run in linear time
- Inference is Modus Ponens (for Horn Form): sound for Horn KB

$$\frac{\alpha_1, \ldots, \alpha_k; \bigwedge_j \alpha_j \Rightarrow \beta}{\beta}$$

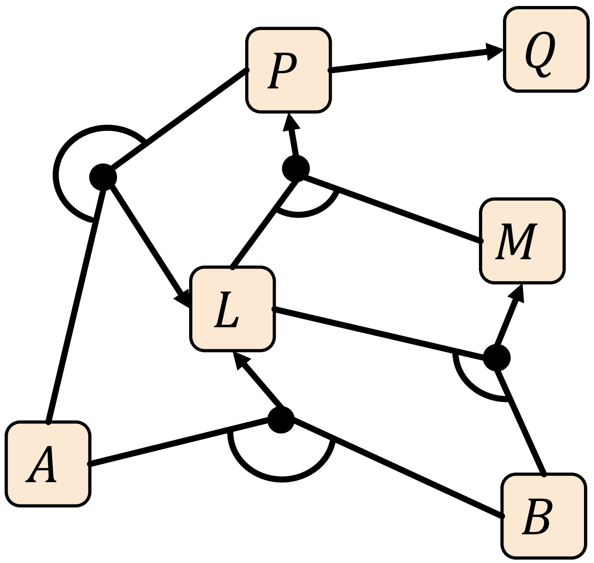
Forward Chaining (FC)

- Idea: Fire any rule whose premise is satisfied in the KB, add its conclusion to the KB, repeat until query is found

KB of horn clauses

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

AND-OR graph



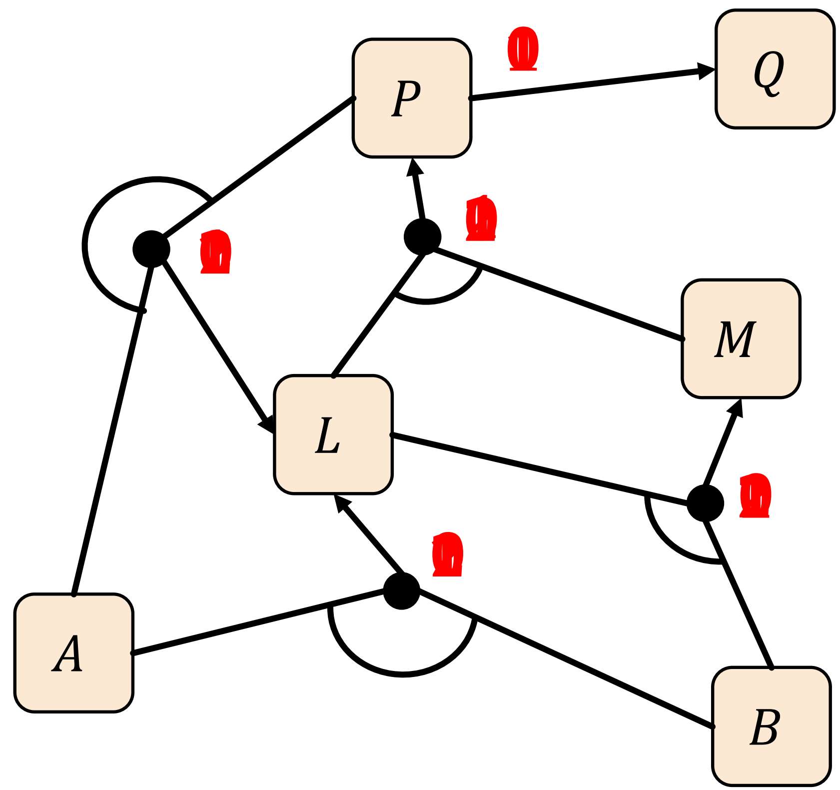
Forward Chaining (FC) Algorithm

- For every rule c, let $\mathrm{count}(c)$ be the number of symbols in c's premise.
- For every symbol s, let $\mathrm{inferred}(s)$ be initially $False$
- Let agenda be a queue of symbols (initially containing all symbols known to be true.
- While $\mathrm{agenda} \neq \emptyset$:
  - pop a symbol p from $agenda$; if it is q we're done
  - Set $inferred(p) = True$
  - For each clause $c \in KB$ such that p is in the premise of c, decrement $count(c)$. If $count(c) = 0$, add c's conclusion to $agenda$.

Forward chaining is sound and complete for Horn KB

Forward Chaining Example

Iteration 1: $[A, B]$
Iteration 2: $[B]$
Iteration 3: $[] \Rightarrow [L]$
Iteration 4: $[] \Rightarrow [M]$
Iteration 5: $[] \Rightarrow [P]$
Iteration 6: $[] \Rightarrow [L, Q]$
Iteration 7: $[Q]$
Iteration 8: $[]$



Proof of Completeness

FC derives every atomic sentence entailed by Horn KB

1. Suppose FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model m that assigns true/false to symbols based on the inferred table
3. Every clause in the original KB is true in m

$$\alpha_1 \wedge \cdots \wedge \alpha_k \Rightarrow \beta$$

4. Hence, m is a model of KB
5. If $KB \vDash q$, then q is true in every model of KB, including m.

Backward Chaining (BC)

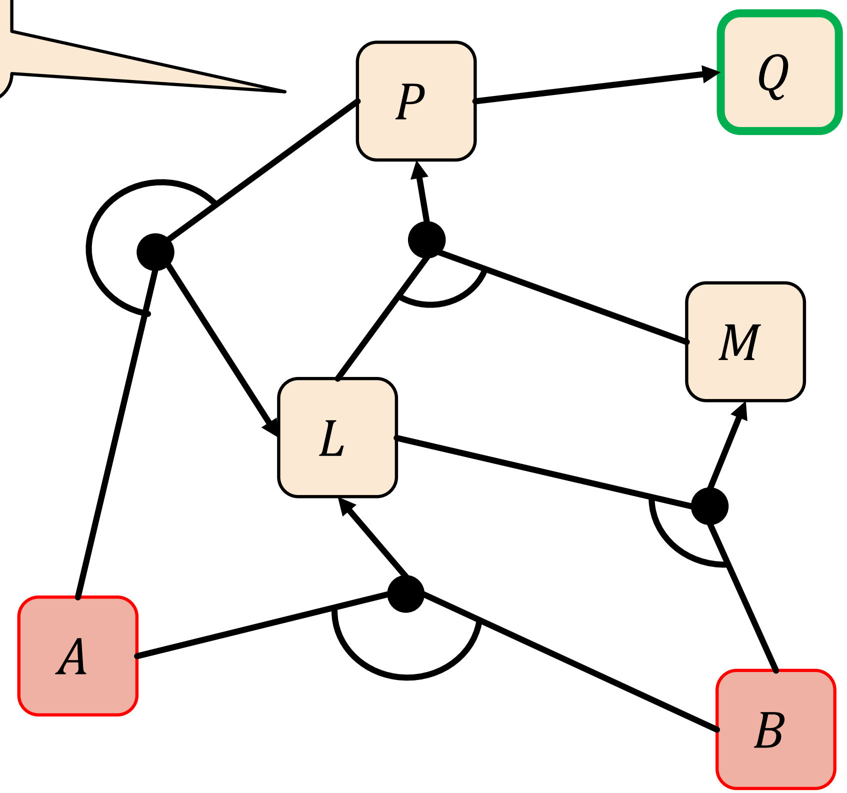
Backtracking depth-first search algorithm

Idea: work backwards from the query q

- To prove q by BC,
  - check if q is known already, or
  - prove by BC the premise of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - has already been proven true, or
  - has already failed

Backwards Chaining Example

Hit a loop! Try something else



Forward vs. Backward Chaining

FC = data-driven reasoning

- e.g., object recognition, routine decisions
- May do a lot of work that is irrelevant to the goal

BC = goal-driven reasoning

- e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be sublinear in $|KB|$.

Proof Methods

Applying inference rules (aka theorem proving)

- Generation of new sentences from old
- Proof = sequential application of inference rules
- Inference rules are actions in a search algorithm
- Proof can be more efficient: ignores irrelevant propositions
- Transformation of sentences into a normal form

Model checking

- Truth table enumeration (time complexity exponential in n)
- Improved backtracking, e.g. DPLL
- local search in model space (sound but incomplete), e.g. min-conflicts-like hill-climbing algorithm

Efficient Propositional Model Checking

Two families of efficient algorithms for propositional model checking:

- Complete backtracking search algorithms
  - DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Incomplete local search algorithms
  - WALKSAT algorithm

These algorithms test a sentence for satisfiability; used for inference.

Recall: Satisfiability is connected to entailment via

$KB \vDash \alpha$ if and only if $(KB \land \neg\alpha)$ is unsatisfiable

DPLL Algorithm

How are DPLL and CSP related?

Determine if a given CNF formula $\phi = C_1 \wedge \cdots \wedge C_m$ is satisfiable

Improvements over truth table enumeration:

1. Early termination

   (a) A clause is true iff any literal in it is true.

   (b) The formula ϕ is false if any clause is false.

2. Pure symbol heuristic

   Least constraining value

   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's literal true: Doing this can never make a clause false.

   Ignore clauses that are already true in the model constructed so far.

3. Unit clause heuristic

   Most constrained variable

   Unit clause: only one literal in the clause.

   The only literal in a unit clause must be true.

DPLL Algorithm

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

  $clauses \leftarrow$ the set of clauses in the CNF representation of s
  $symbols \leftarrow$ a list of the proposition symbols in s
  return DPLL($clauses, symbols, \{\,\}$)

---

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

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

Early Termination

Try to apply heuristics

If it doesn't work, brute force.

WALKSAT Algorithm

- Incomplete, local search algorithm
- Evaluation function: minimize the number of unsatisfied clauses
- Balance between greediness and randomness

WALKSAT Algorithm

CNF formula: $\phi = C_1 \wedge \cdots \wedge C_m$

1. Start with a random variable assignment $\ell_1 \ldots \ell_n$, where $\ell_i \in \{True, False\}$
2. If $\vec{\ell}$ satisfies the formula return $\vec{\ell}$.
3. Choose a random unsatisfied clause $C_j \in \phi$
4. With probability p flip the truth value of a random symbol $x_i \in C_j$; else flip a symbol $x_i \in C_j$ that maximizes number of satisfied clauses in ϕ.
5. Repeat steps 2-4 $MaxFlips$ times.

Why is WalkSat incomplete?

How are WALKSAT and local search related?

Inference-Based Agents in the Wumpus World

A wumpus-world agent using propositional logic:

$\neg P_{1,1}$

$\neg W_{1,1}$

$B_{i,j} \Leftrightarrow \left(P_{i,j+1} \vee P_{i,j-1} \vee P_{i+1,j} \vee P_{i-1,j}\right)$

$S_{i,j} \Leftrightarrow \left(W_{i,j+1} \vee W_{i,j-1} \vee W_{i+1,j} \vee W_{i-1,j}\right)$

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

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

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

...

Each i, j rule is its own proposition

There is exactly one Wumpus

64 distinct proposition symbols, 155 sentences

Expressiveness Limitation of Propositional Logic

- KB contains "physics" sentences for every single square
- For every time t and every location $[i, j]$,

  $L_{i,j}^{t} \wedge FacingEast^{t} \wedge Forward^{t} \Rightarrow L_{i+1,j}^{t}$

Rapid proliferation of clauses