

CS 188: Artificial Intelligence

Spring 2007

Lecture 8: Logical Agents - I

2/8/2007

Srini Narayanan – ICSI and UC Berkeley

Many slides over the course adapted from Dan Klein, Stuart Russell or Andrew Moore

Announcements

- § Concurrent Enrollment
- § Assignment 1 Solutions up
- § Note on notational variants

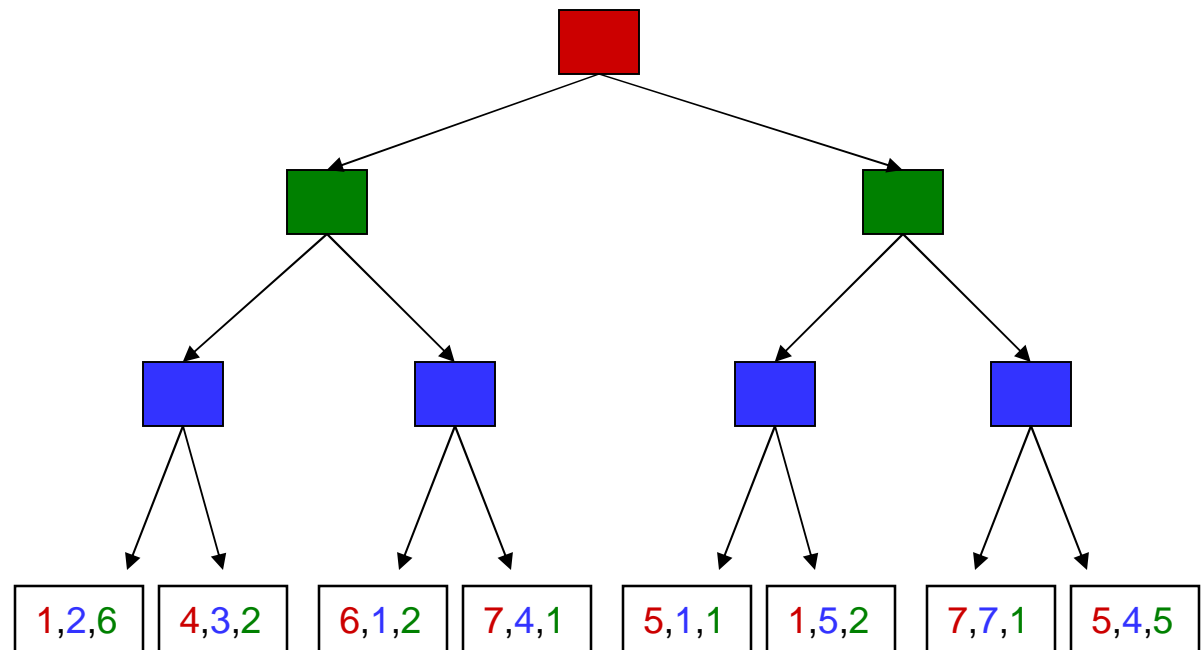
Non-Zero-Sum Games

§ Similar to
minimax:

§ Utilities are
now tuples

§ Each player
maximizes
their own entry
at each node

§ Propagate (or
back up) nodes
from children

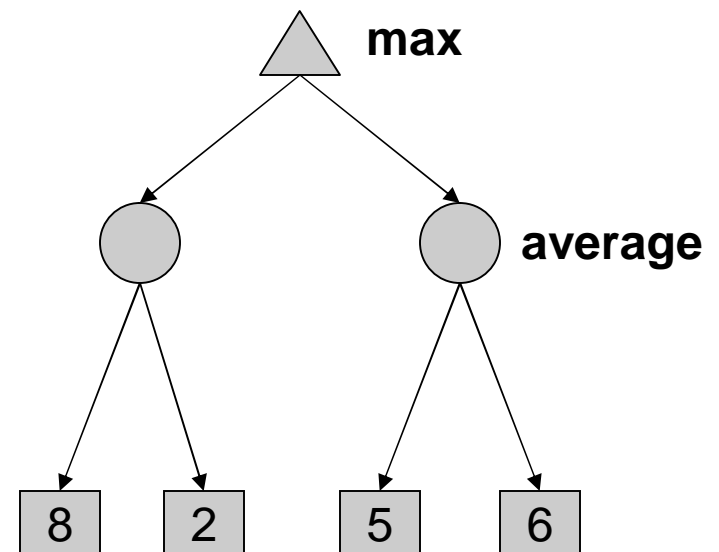


Stochastic Single-Player

- § What if we don't know what the result of an action will be? E.g.,
 - § In solitaire, shuffle is unknown
 - § In minesweeper, don't know where the mines are

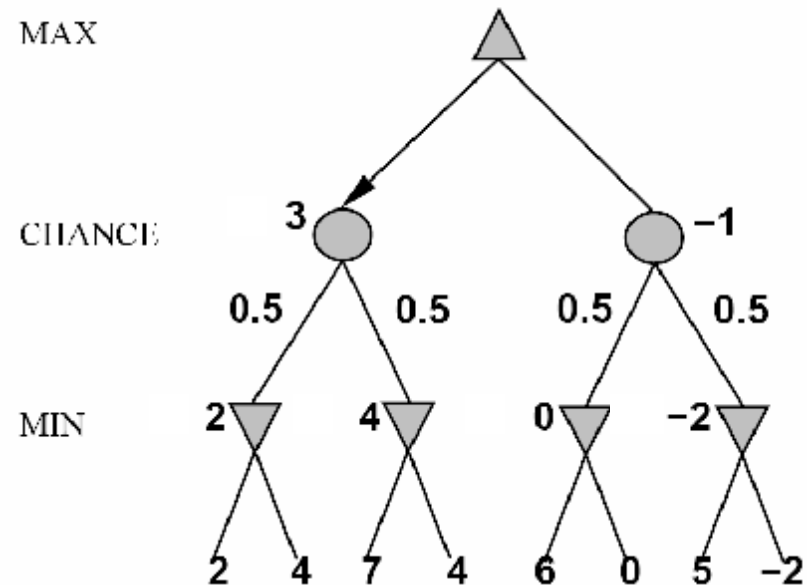
- § Can do **expectimax search**
 - § Chance nodes, like actions except the environment controls the action chosen
 - § Calculate utility for each node
 - § Max nodes as in search
 - § Chance nodes take average (expectation) of value of children

- § Later, we'll learn how to formalize this as a **Markov Decision Process**



Stochastic Two-Player

- § E.g. backgammon
- § Expectiminimax (!)
- § Environment is an extra player that moves after each agent
- § Chance nodes take expectations, otherwise like minimax



if *state* is a MAX node then

return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)

if *state* is a MIN node then

return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)

if *state* is a chance node then

return average of EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)

Game Playing State-of-the-Art

- § **Checkers:** Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.
- § **Chess:** Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue examined 200 million positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply.
- § **Othello:** human champions refuse to compete against computers, which are too good.
- § **Go:** human champions refuse to compete against computers, which are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.

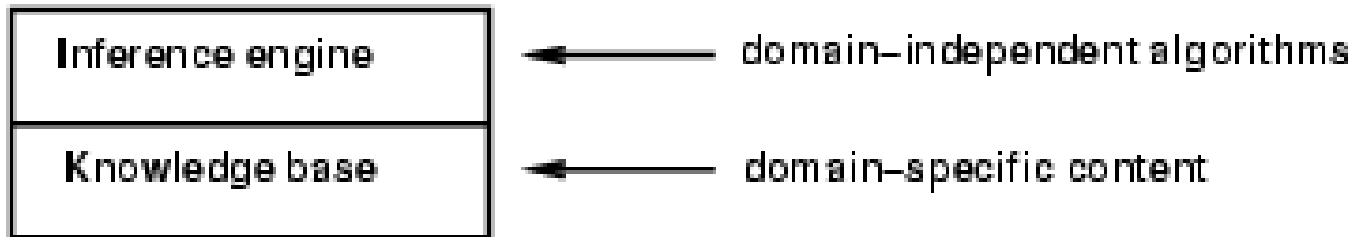
Logical Agents

- § Reflex agents find their way from Arad to Bucharest by dumb luck.
- § Chess program calculates legal moves of its king, but doesn't know that no piece can be on 2 different squares at the same time
- § *Logic (Knowledge-Based) agents* combine
 - § **general knowledge &**
 - § **current percepts**
 - § **to *infer* hidden aspects of current state prior to selecting actions**
 - § ***Crucial in partially observable environments***

Outline

- § Knowledge-based agents
- § Wumpus world example
- § Logic in general - models and entailment
- § Propositional (Boolean) logic
- § Equivalence, validity, satisfiability
- § Inference

Knowledge bases



- § 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., what they know, regardless of how implemented
- § Or at the implementation level
 - § i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
         t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-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 representations of the world
 - § Deduce hidden properties of the world
 - § Deduce appropriate actions

Wumpus World PEAS description

§ Performance measure

§ gold +1000, death -1000

§ -1 per step, -10 for using the arrow

§ Environment

§ Squares adjacent to wumpus are smelly

§ Squares adjacent to pit are breezy

§ Glitter iff gold is in the same square

§ Shooting kills wumpus if you are facing it

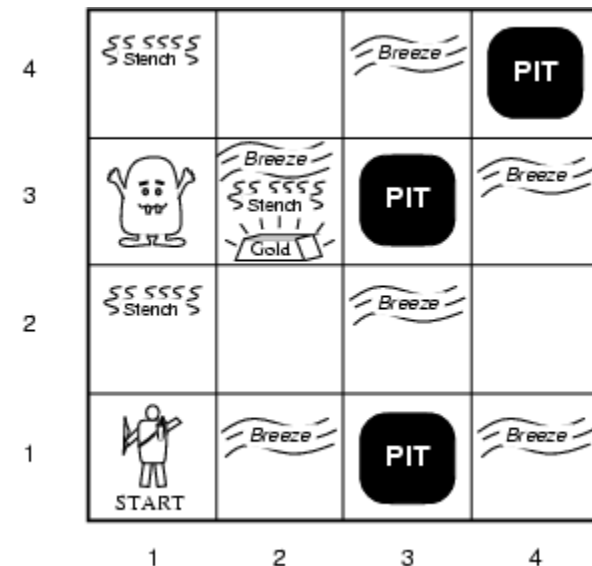
§ Shooting uses up the only arrow

§ Grabbing picks up gold if in same square

§ Releasing drops the gold in same square

§ Sensors: Stench, Breeze, Glitter, Bump, Scream

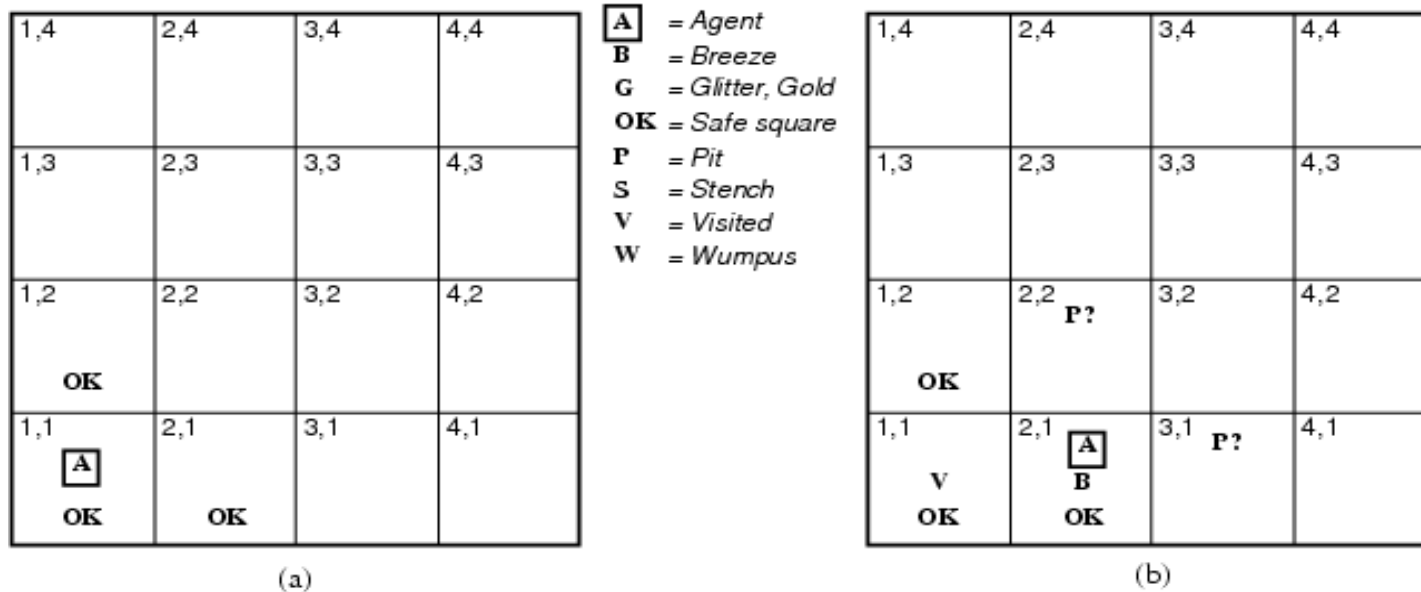
§ Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot



Wumpus world characterization

- § Fully Observable No – only local perception
- § Deterministic Yes – outcomes exactly specified
- § Episodic No – sequential at the level of actions
- § Static Yes – Wumpus and Pits do not move
- § Discrete Yes
- § Single-agent? Yes – Wumpus is essentially a natural feature

Exploring the Wumpus World



1. The KB initially contains the rules of the environment.
2. [1,1] The first percept is *[none, none, none, none, none]*,
Move to safe cell e.g. 2,1
3. [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

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

(a)

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

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

(b)

- [1,2] Stench in cell: wumpus is in [1,3] or [2,2]
YET ... not in [1,1]
Thus ... not in [2,2] or stench would have been detected in [2,1]
Thus ... wumpus is in [1,3]
Thus ... [2,2] is safe because of lack of breeze in [1,2]
Thus ... pit in [3,1]
 Move to next safe cell [2,2]

Exploring the Wumpus World

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

(a)

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

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

(b)

5. [2,2] Detect nothing
 Move to unvisited safe cell e.g. [2,3]
6. [2,3] Detect glitter , smell, breeze
 Thus... pick up gold
 Thus... pit in [3,3] or [2,4]

Logic in general

- § Logics are formal languages for representing information such that conclusions can be drawn
- § 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., the language of arithmetic
 - § $x+2 \geq y$ is a sentence; $x^2+y > \{ \}$ is not a sentence
 - § $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
 - § $x+2 \geq y$ 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

§ Entailment means that one thing **follows from** another:

$$KB \models \alpha$$

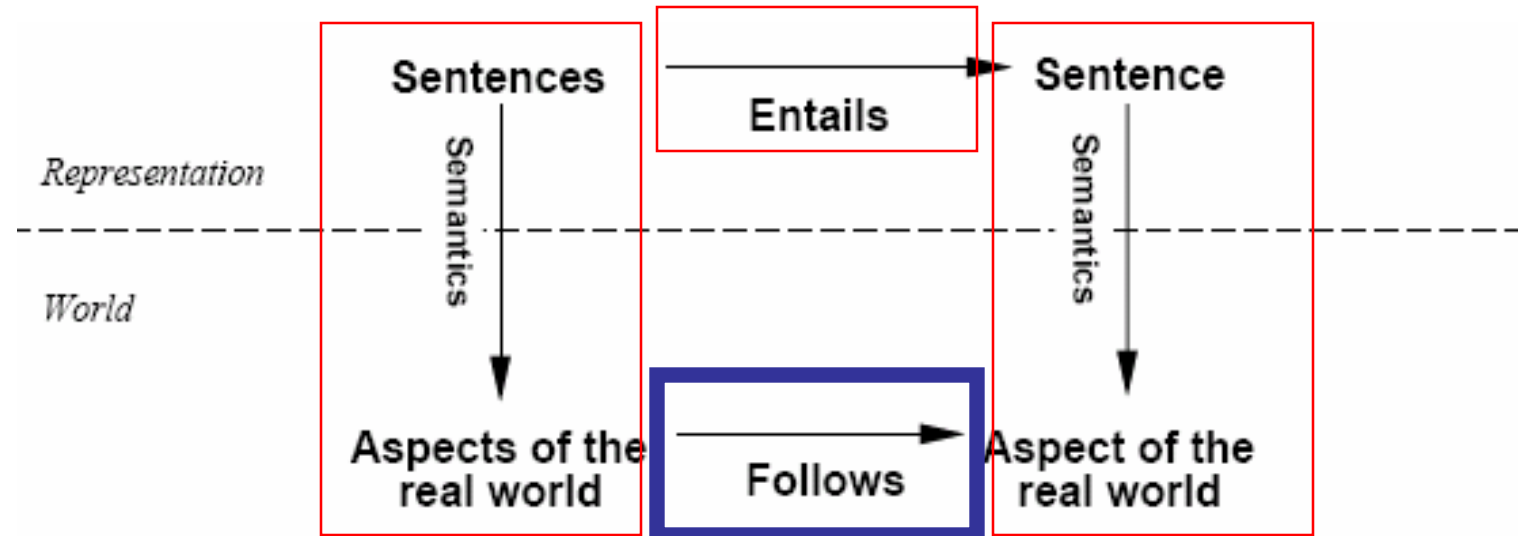
§ Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

§ E.g., the KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”

§ E.g., $x+y = 4$ entails $4 = x+y$

§ Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**

Schematic perspective



If KB is true in the real world, then any sentence a derived from KB by a sound inference procedure is also true in the real world.

Models

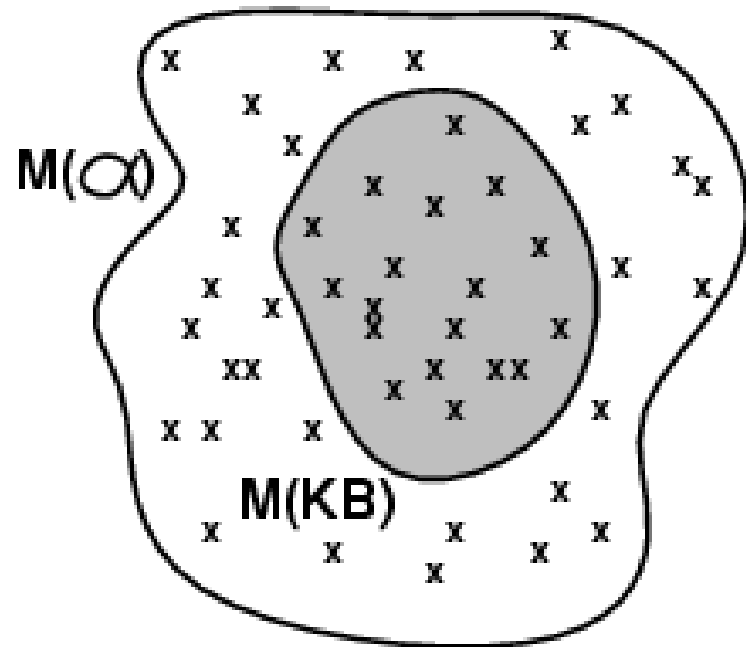
§ Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

§ We say m is a model of a sentence α if α is true in m

§ $M(\alpha)$ is the set of all models of α

§ Then $KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$

§ E.g. $KB =$ Giants won and Reds won
 $\alpha =$ Giants won

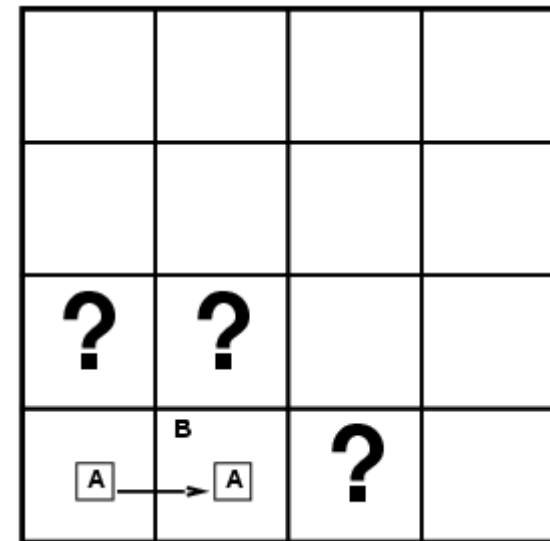


Entailment in the wumpus world

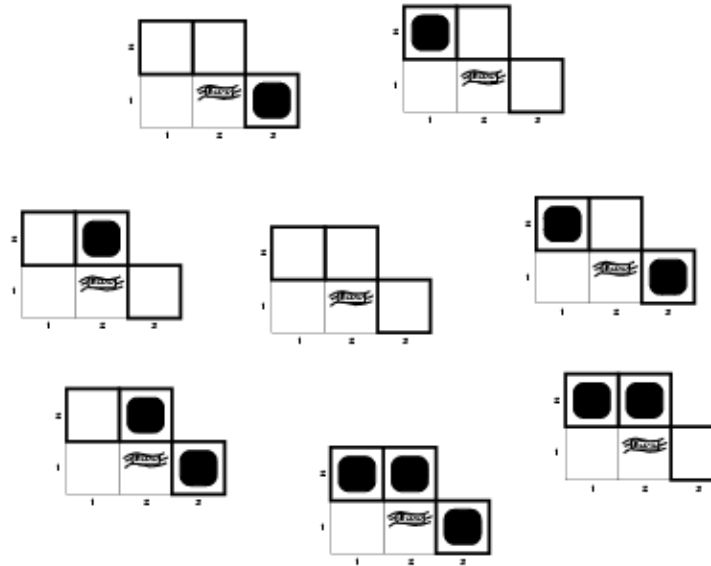
Situation after detecting
nothing in [1,1], moving
right, breeze in [2,1]

Consider possible models for
KB assuming only pits

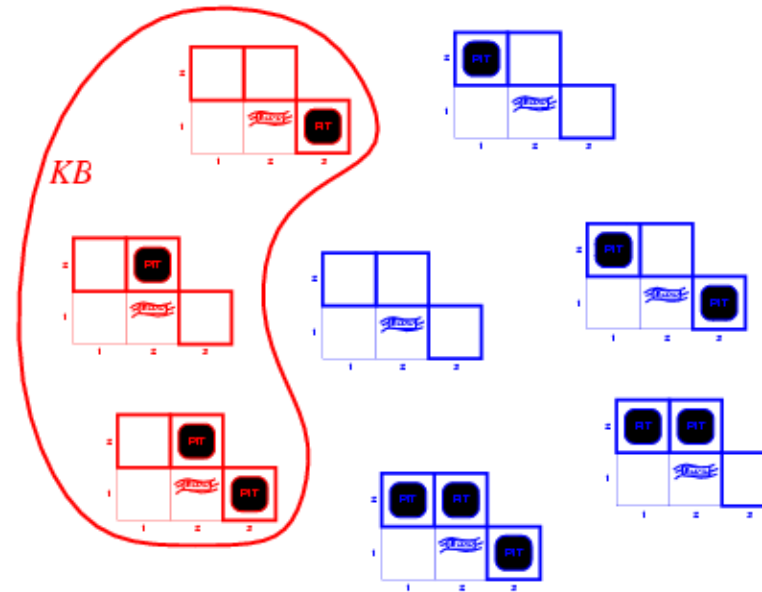
3 Boolean choices \Rightarrow 8
possible models



Wumpus models

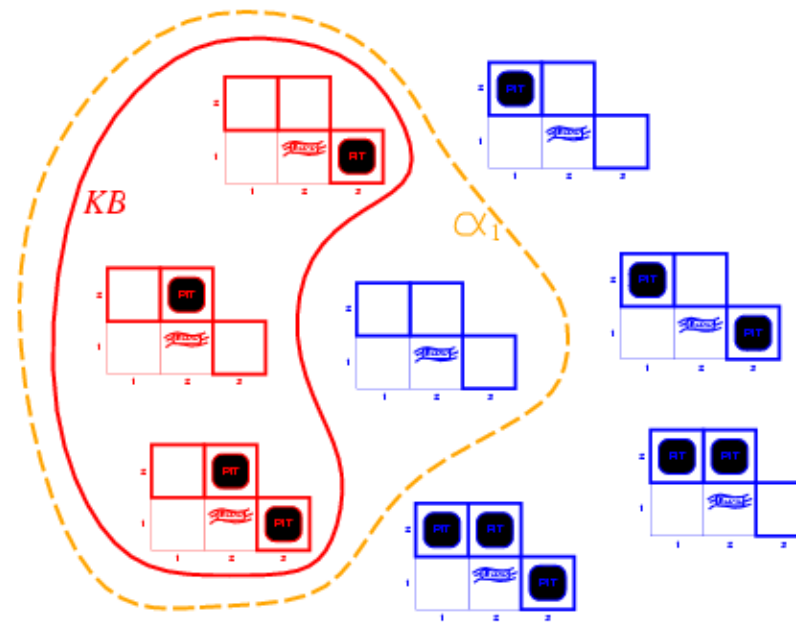


Wumpus models



§ $KB = \text{wumpus-world rules} + \text{observations}$

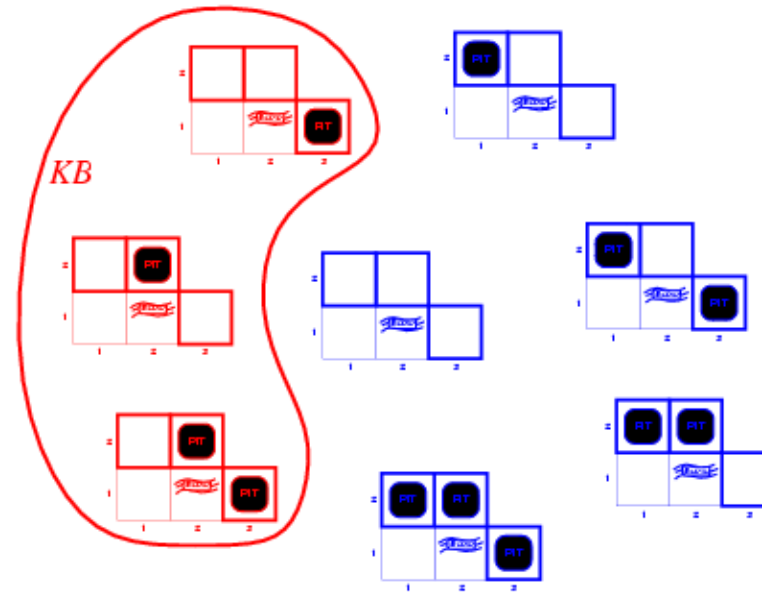
Wumpus models



§ KB = wumpus-world rules + observations

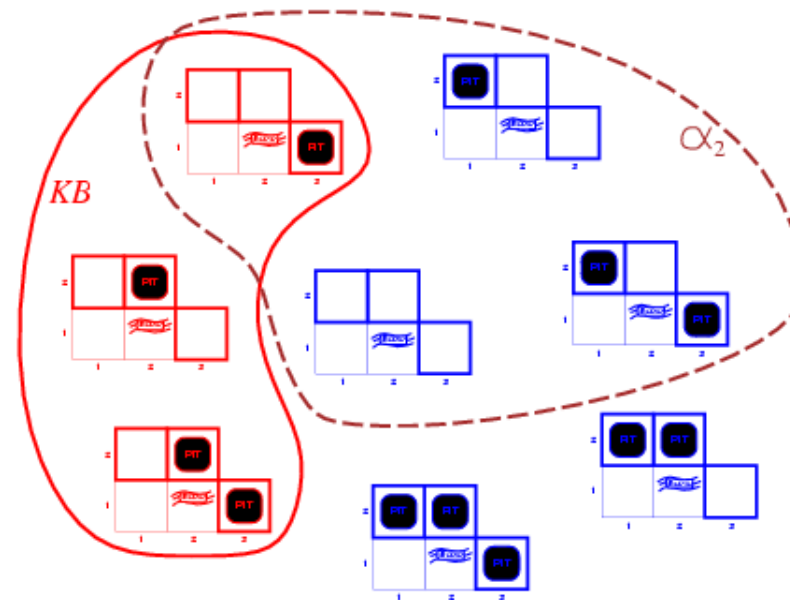
§ α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



§ $KB = \text{wumpus-world rules} + \text{observations}$

Wumpus models



§ KB = wumpus-world rules + observations

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

Inference Procedures

- § $KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i
- § **Soundness**: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
(no wrong inferences but maybe not all true statements can be derived)
- § **Completeness**: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
(all true sentences can be derived, but maybe some wrong extra ones as well)

Propositional logic: Syntax

§ Propositional logic is the simplest logic – illustrates basic ideas

§ The proposition symbols P_1, P_2 etc are 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 specifies true/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.,

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

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

OR: P or Q is true or both are true.
XOR: P or Q is true but not both.

Implication is always true when the premises are False!

Wumpus world sentences

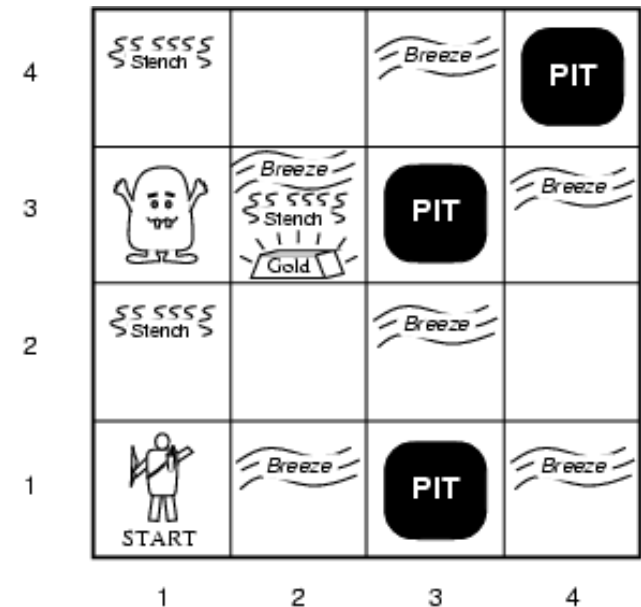
Let $P_{i,j}$ be true if there is a pit in $[i, j]$.

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

start: $\emptyset P_{1,1}$
 $\emptyset B_{1,1}$
 $B_{2,1}$

"Pits cause breezes in adjacent squares"

$B_{1,1} \hat{U} (P_{1,2} \hat{U} P_{2,1})$
 $B_{2,1} \hat{U} (P_{1,1} \hat{U} P_{2,2} \hat{U} P_{3,1})$



Truth tables for inference

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

Inference by enumeration

§ Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB,  $\alpha$ ) returns true or false
  symbols  $\leftarrow$  a list of the proposition symbols in KB and  $\alpha$ 
  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 do
    P  $\leftarrow$  FIRST(symbols); rest  $\leftarrow$  REST(symbols)
    return TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, true, model)) and
      TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, false, model))
```

§ PL-True returns true if the sentence holds within the model

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

Validity and satisfiability

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

e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

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

A sentence is unsatisfiable if it is true in no 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

Satisfiability of propositional logic was instrumental in developing the theory of NP-completeness.

Proof methods

§ Proof methods divide into (roughly) two kinds:

§ Application of inference rules

§ Legitimate (sound) generation of new sentences from old

§ Proof = a sequence of inference rule applications

Can use inference rules as operators in a standard search algorithm

§ Typically require transformation of sentences into a normal form

§ Model checking

§ 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

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 $\beta \models \alpha$

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$

$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$

$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$

$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$

$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$

$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$


$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$

$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$

$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$

$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$

$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$



You need to know these !

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:

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

4. Apply distributivity 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

Conjunctive Normal Form (CNF)

conjunction of disjunctions of literals

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$:

Basic intuition, resolve $B, \neg B$ to get $(A) \vee (\neg C \vee \neg D)$ (why?)

§ Resolution inference rule (for CNF):

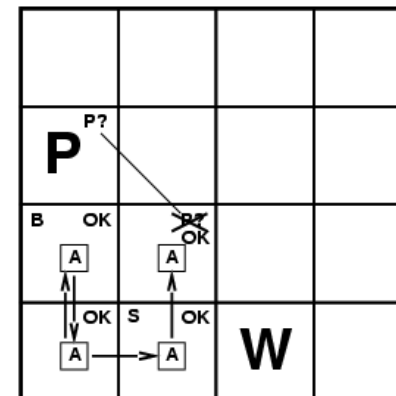
$$\frac{l_i \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{l_i \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

where l_i and m_j are complementary literals.

$$\text{E.g., } \frac{P_{1,3} \vee P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}$$

§ Resolution is sound and complete for propositional logic.

§ Basic Use: $KB \models \alpha$ iff $(KB \wedge \neg \alpha)$ is unsatisfiable



Resolution

Soundness of resolution inference rule:

$$\frac{\begin{array}{l} \neg(l_i \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k) \Rightarrow l_i \\ \neg m_j \Rightarrow (m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n) \end{array}}{\neg(l_i \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k) \Rightarrow (m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n)}$$

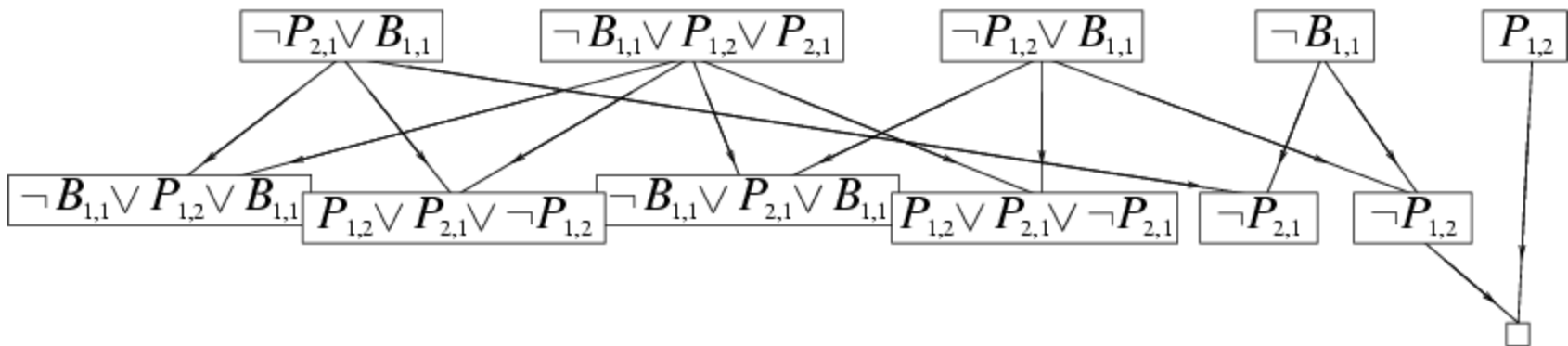
Resolution algorithm

§ Proof by contradiction, i.e., show $KB \wedge \neg \alpha$ unsatisfiable

```
function PL-RESOLUTION( $KB, \alpha$ ) returns true or false
   $clauses \leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg \alpha$ 
   $new \leftarrow \{ \}$ 
  loop do
    for each  $C_i, C_j$  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
   $clauses \leftarrow clauses \cup new$ 
```

Resolution example

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



Efficient propositional inference

Two families of efficient algorithms for propositional inference:

Complete backtracking search algorithms

§ DPLL algorithm (Davis, Putnam, Logemann, Loveland)

§ Incomplete local search algorithms

§ WalkSAT algorithm

The DPLL algorithm

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. Early termination

A clause is true if any literal is true.

A sentence is false if any clause is false.

2. Pure symbol heuristic

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

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

Make a pure symbol literal true.

3. Unit clause heuristic

Unit clause: only one literal in the clause

The only literal in a unit clause must be true.

The WalkSAT algorithm

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

The WalkSAT algorithm

```
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.,

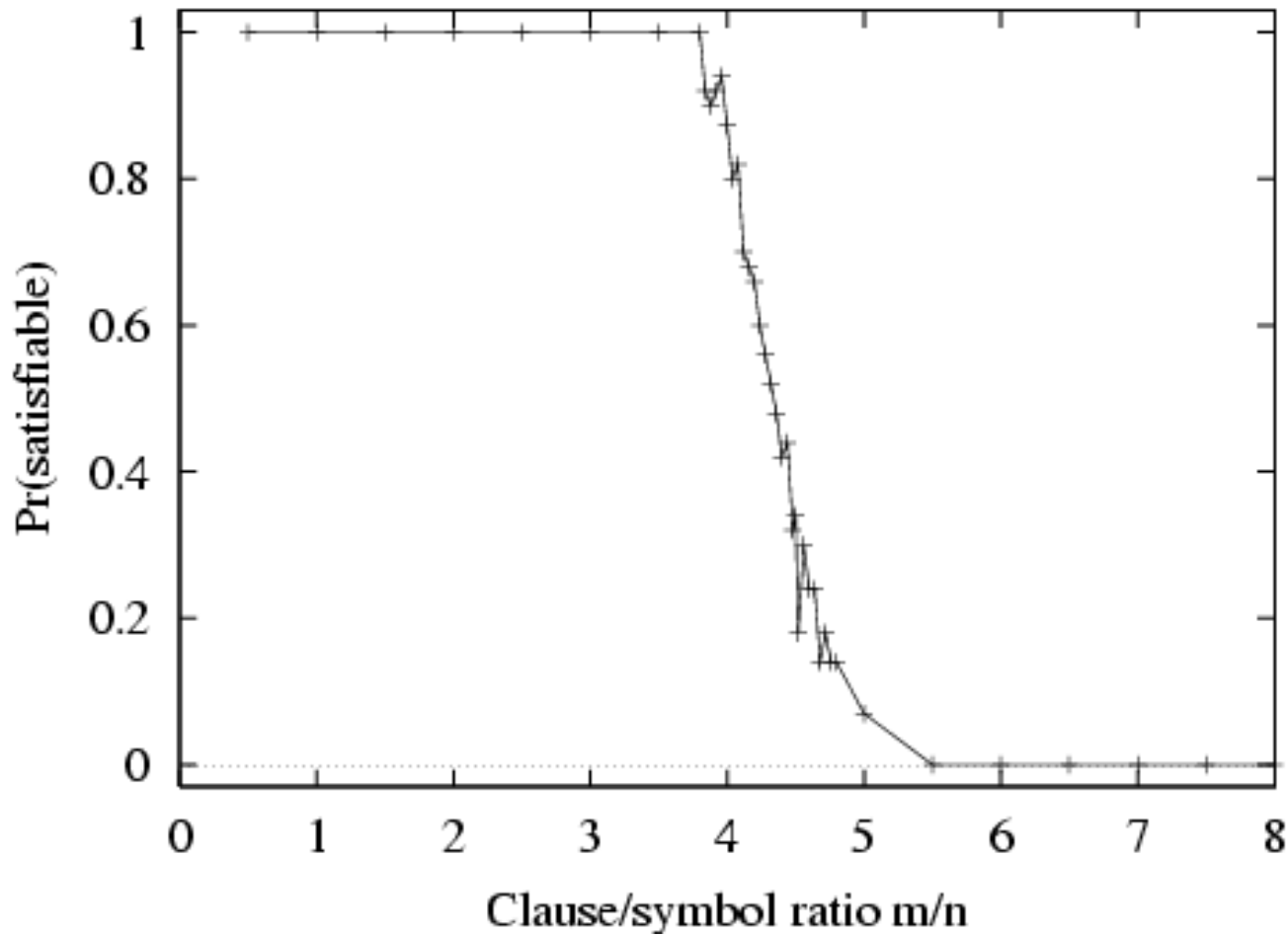
$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

m = number of clauses

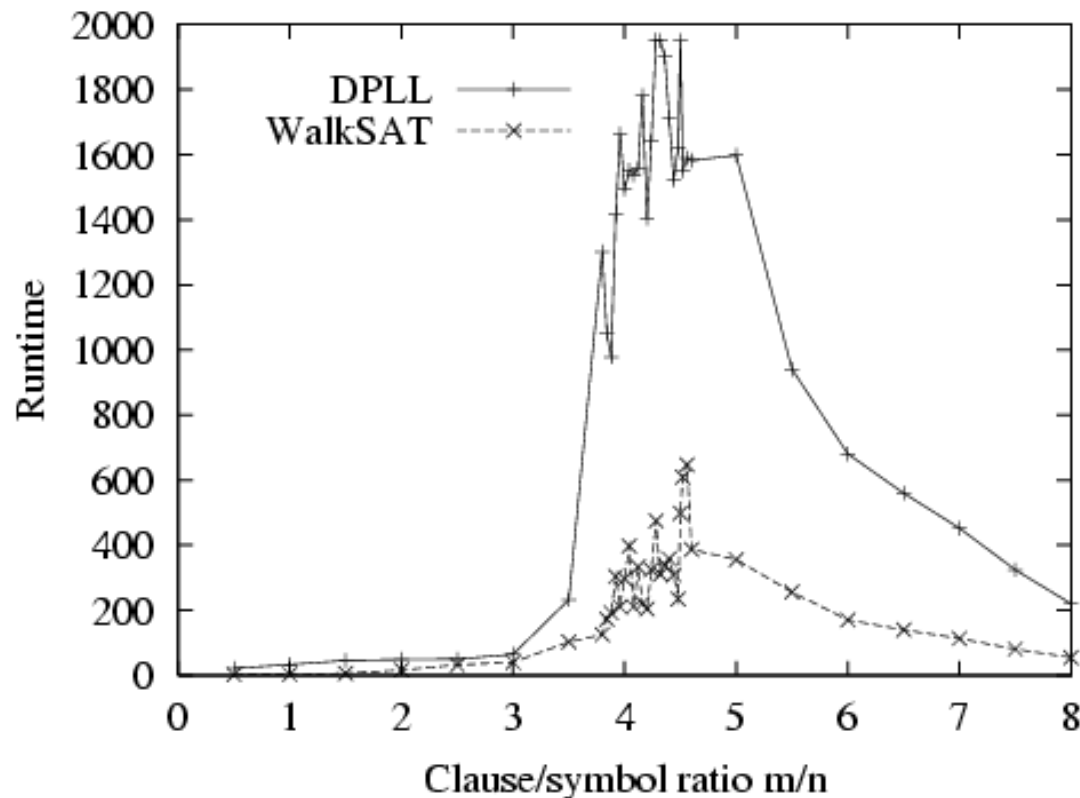
n = number of symbols

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

Hard satisfiability problems



Hard satisfiability problems



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

Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

$$\begin{aligned} & \neg P_{1,1} \\ & \neg W_{1,1} \\ & B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y}) \\ & S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y}) \\ & W_{1,1} \vee W_{1,2} \vee \dots \vee W_{4,4} \\ & \neg W_{1,1} \vee \neg W_{1,2} \\ & \neg W_{1,1} \vee \neg W_{1,3} \\ & \dots \end{aligned}$$

⇒ 64 distinct proposition symbols, 155 sentences


```

function PL-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench, breeze, glitter]
  static: KB, initially containing the “physics” of the wumpus world
           x, y, orientation, the agent’s position (init. [1,1]) and orient. (init. right)
           visited, an array indicating which squares have been visited, initially false
           action, the agent’s most recent action, initially null
           plan, an action sequence, initially empty

  update x, y, orientation, visited based on action
  if stench then TELL(KB,  $S_{x,y}$ ) else TELL(KB,  $\neg S_{x,y}$ )
  if breeze then TELL(KB,  $B_{x,y}$ ) else TELL(KB,  $\neg B_{x,y}$ )
  if glitter then action  $\leftarrow$  grab
  else if plan is nonempty then action  $\leftarrow$  POP(plan)
  else if for some fringe square [i, j], ASK(KB, ( $\neg P_{i,j} \wedge \neg W_{i,j}$ )) is true or
           for some fringe square [i, j], ASK(KB, ( $P_{i,j} \vee W_{i,j}$ )) is false then do
           plan  $\leftarrow$  A*-GRAPH-SEARCH(ROUTE-PB([x, y], orientation, [i, j], visited))
           action  $\leftarrow$  POP(plan)
  else action  $\leftarrow$  a randomly chosen move
  return action

```

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
- § Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- § Resolution is complete for propositional logic
Forward, backward chaining are linear-time, complete for Horn clauses
- § Propositional logic lacks expressive power