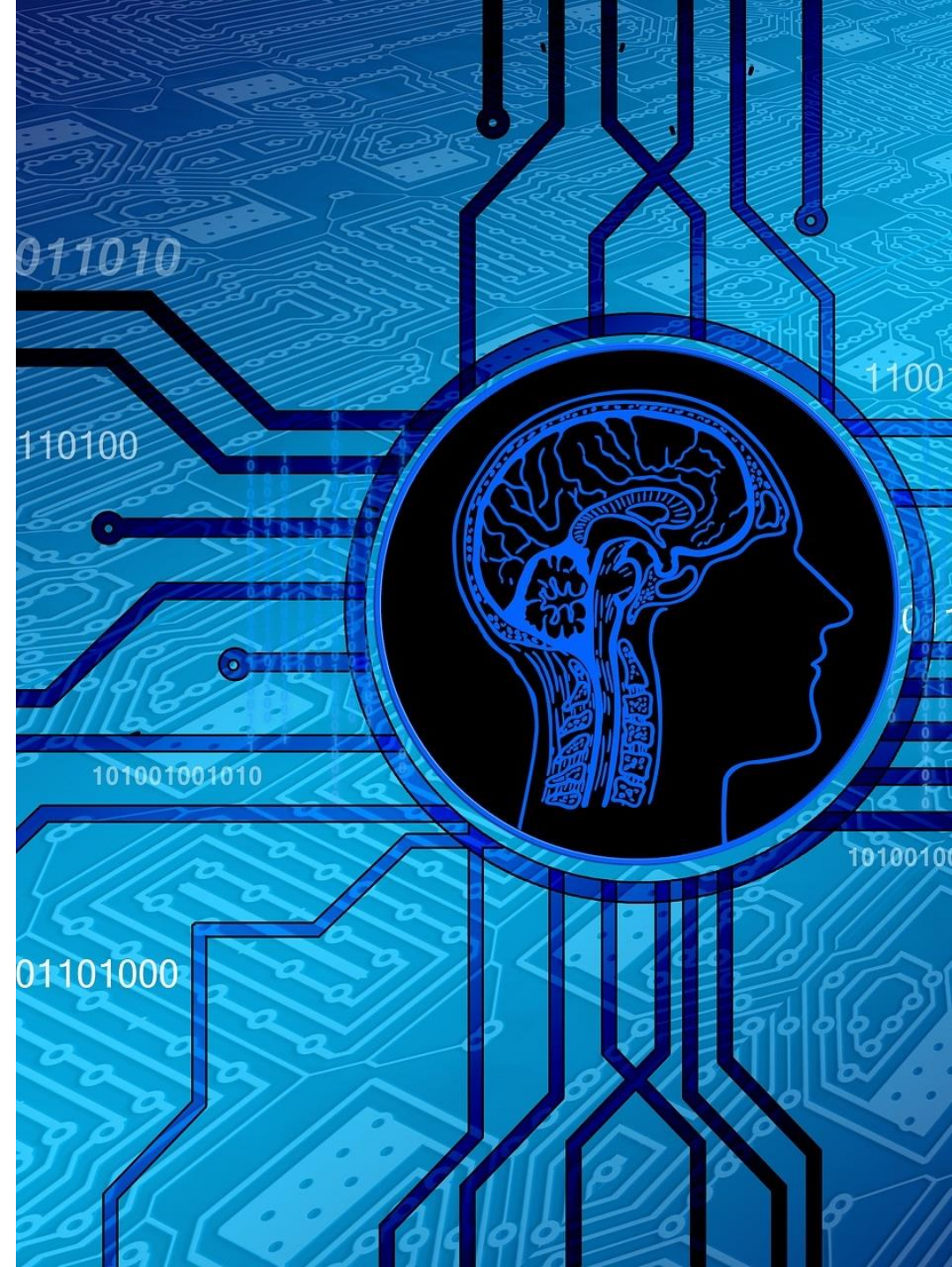


Effective Propositional Inference

Informatics 2D: Reasoning and Agents



Outline

Two families of **efficient** algorithms for propositional inference:

Complete backtracking search algorithms

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

Incomplete local search algorithms

- WalkSAT algorithm

Clausal Form (CNF)

DPLL and WalkSAT manipulate formulae in **conjunctive normal form (CNF)**.

Sentence

- Formula whose satisfiability is to be determined
- Conjunction of clauses

Clause

- Disjunction of literals

Literal

- Proposition symbol or negated proposition symbol

e.g. $(A, \neg B), (B, \neg C)$ represents $(A \vee \neg B) \wedge (B \vee \neg C)$

Recall Logical Equivalence

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of \wedge
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of \vee
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of \wedge
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	associativity of \vee
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \rightarrow \beta) \equiv (\neg\beta \rightarrow \neg\alpha)$	contraposition
$(\alpha \rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \leftrightarrow \beta) \equiv ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of \wedge over \vee
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of \vee over \wedge

Two sentences are **logically equivalent** iff **true** in the same models:

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

Conversion to CNF

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

Eliminate \Leftrightarrow : replace $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

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

Eliminate \Rightarrow : replace $\alpha \Rightarrow \beta$ with $\neg \alpha \vee \beta$

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

Move \neg inwards : use de Morgan's rules and double negation $\neg \neg \alpha = \alpha$

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

Create clauses: apply distributivity law (\vee over \wedge) and flatten

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

DPLL

The DPLL algorithm

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

Improvements over truth table enumeration:

1. Early termination
2. Pure symbol heuristic
3. Unit clause heuristic

1. Early termination

- A *clause* is true if **one** of its literals is true,
 - e.g., if A is true then $(A \vee \neg B)$ is true.
- A *sentence* is false if **any** of its clauses is false,
 - e.g., if A is false and B is true then
 - $(A \vee \neg B)$ is **false**, so any sentence containing it is **false**.

2. Pure symbol heuristic

- **Pure symbol:** **always** appears with the same "*sign*"/*polarity* 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 **literal** containing a pure symbol true.
 - e.g., Let A and $\neg B$ both be true.

3. Unit clause heuristic

- **Unit clause**: only one literal in the clause
 - e.g. (A)
- The only literal in a unit clause must be true.
 - e.g., A must be true.
- Also includes clauses where **all but one** literal is false,
 - e.g. (A,B,C) where B and C are false since it is equivalent to (A, false, false) i.e. (A).

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* })

P, *value* \leftarrow FIND-UNIT-CLAUSE(*clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols* – *P*, *model* \cup { *P*=*value* })

P \leftarrow FIRST(*symbols*); *rest* \leftarrow REST(*symbols*)

return DPLL(*clauses*, *rest*, *model* \cup { *P*=*true* }) **or**
 DPLL(*clauses*, *rest*, *model* \cup { *P*=*false* })

The DPLL algorithm

Tautology Deletion (Optional)

- **Tautology**: both a proposition and its negation in a clause.
 - e.g. $(A, B, \neg A)$
- Clause bound to be true.
 - e.g., whether A is true or false.
 - Therefore, can be **deleted**.

Mid-Lecture Exercise

- Apply DPLL heuristics to the following sentence:

$$\begin{aligned} & (S_{2,1}), (\neg S_{1,1}), (\neg S_{1,2}), \\ & (\neg S_{2,1}, W_{2,2}), (\neg S_{1,1}, W_{2,2}), (\neg S_{1,2}, W_{2,2}), \\ & (\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2}) \end{aligned}$$

- Use **case splits** if model not found by the heuristics.
- Symbols: $S_{1,1}, S_{1,2}, S_{2,1}, W_{2,2}$

Solution

Pure symbol heuristic:

$(S_{2,1})$

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

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

$(\neg S_{2,1}, W_{2,2})$

$(\neg S_{1,1}, W_{2,2})$

$(\neg S_{1,2}, W_{2,2})$

$(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$

Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

$(S_{2,1})$

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

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

$(\neg S_{2,1}, W_{2,2})$

$(\neg S_{1,1}, W_{2,2})$

$(\neg S_{1,2}, W_{2,2})$

$(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$

Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

- $S_{2,1}$ is true

T

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

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

(F, W_{2,2})

$(\neg S_{1,1}, W_{2,2})$

$(\neg S_{1,2}, W_{2,2})$

$(\neg W_{2,2}, \mathbf{T}, S_{1,1}, S_{1,2})$

Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

- $S_{2,1}$ is true

Early termination heuristic:

- $(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$ is true

T

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

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

(F, W_{2,2})

$(\neg S_{1,1}, W_{2,2})$

$(\neg S_{1,2}, W_{2,2})$

T

Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

- $S_{2,1}$ is true
- $S_{1,1}$ is false

Early termination heuristic:

- $(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$ is true

T

T

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

(F, $W_{2,2}$)

(T, $W_{2,2}$)

$(\neg S_{1,2}, W_{2,2})$

T

Solution

Pure symbol heuristic:

- No literal is pure.

T

Unit clause heuristic:

- $S_{2,1}$ is true
- $S_{1,1}$ is false
- $S_{1,2}$ is false

T

T

(F, $W_{2,2}$)

T

Early termination heuristic:

- $(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$ is true
- $(\neg S_{1,1}, W_{2,2})$ is true

(T, $W_{2,2}$)

T

Solution

Pure symbol heuristic:

- No literal is pure.

T

Unit clause heuristic:

- $S_{2,1}$ is true
- $S_{1,1}$ is false
- $S_{1,2}$ is false

T

T

(F, $W_{2,2}$)

T

Early termination heuristic:

- $(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$ is true
- $(\neg S_{1,1}, W_{2,2})$ is true
- $(\neg S_{2,1}, W_{2,2})$ is true

T

T

Solution

Pure symbol heuristic:

- No literal is pure.

T

Unit clause heuristic:

- $S_{2,1}$ is true
- $S_{1,1}$ is false
- $S_{1,2}$ is false
- $W_{2,2}$ is true

T

T

T

T

T

Early termination heuristic:

- $(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$ is true
- $(\neg S_{1,1}, W_{2,2})$ is true
- $(\neg S_{2,1}, W_{2,2})$ is true

T

WalkSAT



The WalkSAT algorithm



- **Incomplete**, local search algorithm
- **Evaluation function**:
 - The **min-conflict heuristic** of minimizing the number of unsatisfied clauses
- Algorithm checks for satisfiability by **randomly** flipping the **values** of variables
- Balance between **greediness** and **randomness**

```
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, typically around 0.5  
           max_flips, number of flips allowed before giving up  
  
  model  $\leftarrow$  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  $\leftarrow$  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
```

The WalkSAT algorithm



Hard satisfiability problems

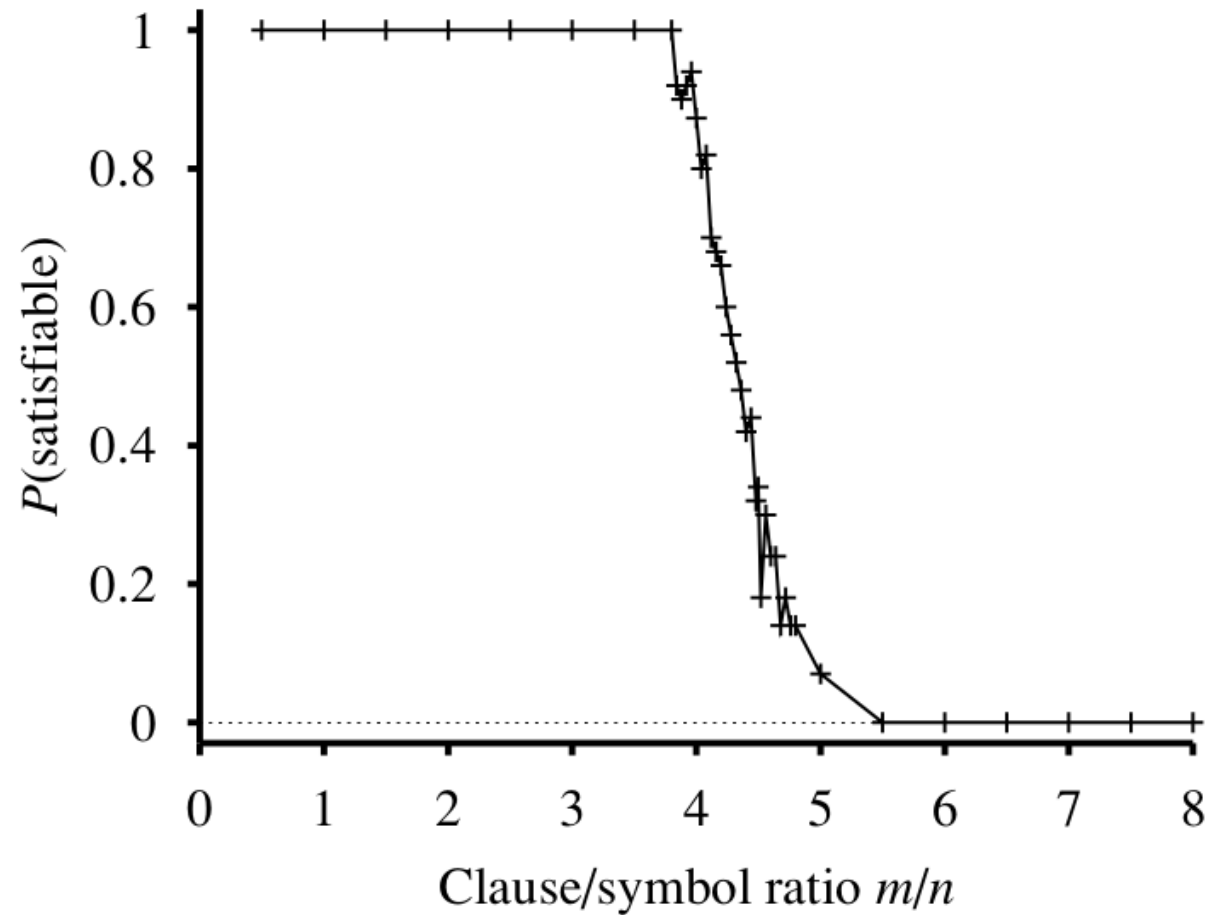
➤ Consider random 3-CNF sentences.

- Example:

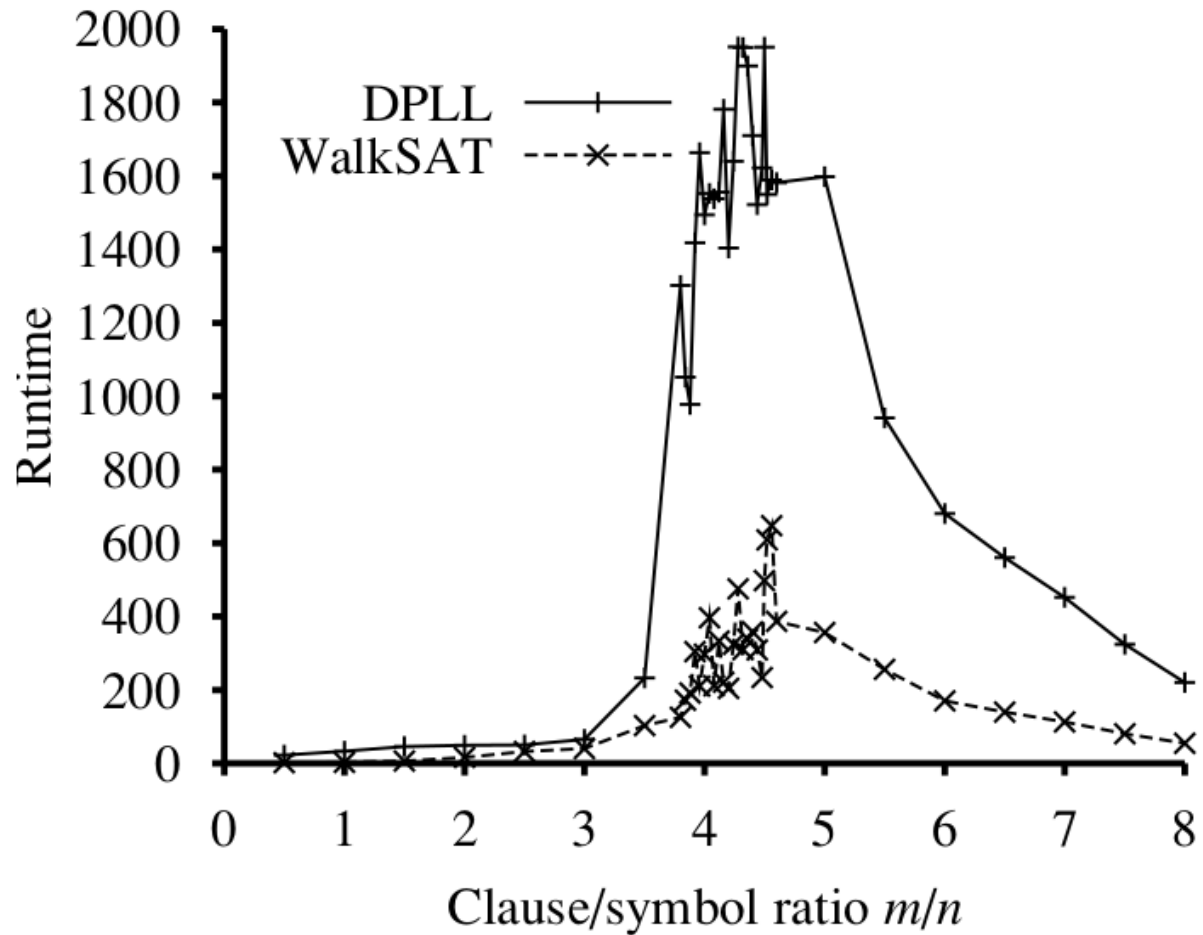
$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

- m = number of clauses
- n = number of symbols

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

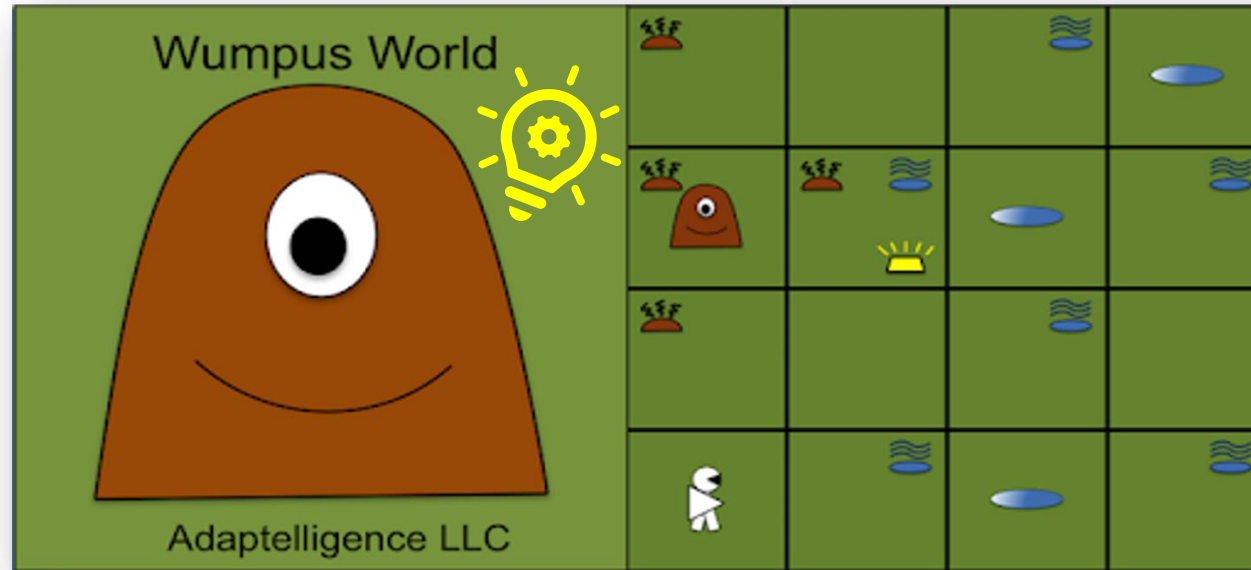


Hard satisfiability problems



Hard satisfiability problems

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



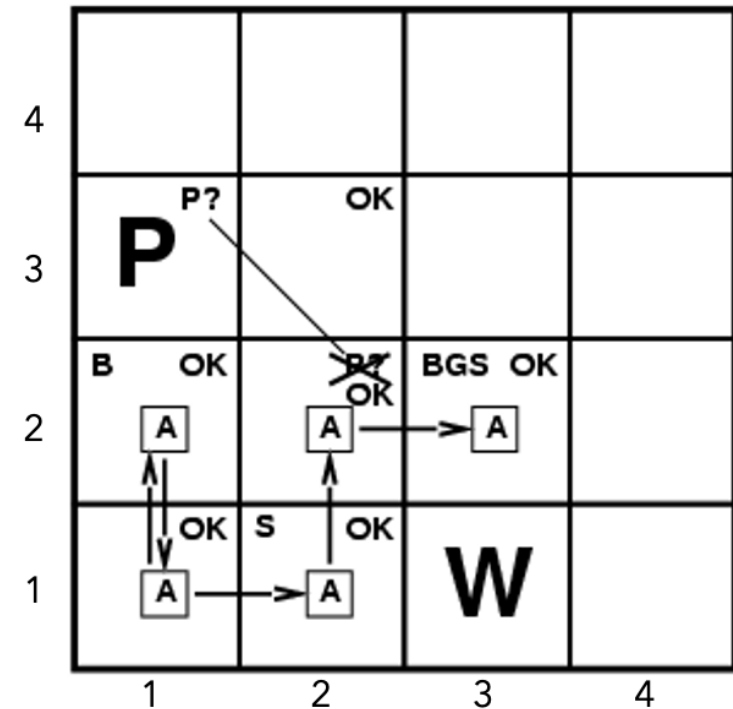
Inference in the Wumpus World

Inference-based agents in the wumpus world

➤ A wumpus-world agent using **propositional logic**:

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

➤ **64** distinct proposition symbols, **155** sentences



```

function HYBRID-WUMPUS-AGENT(percept) returns an action
  inputs: percept, a list, [stench, breeze, glitter, bump, scream]
  persistent: KB, a knowledge base, initially the atemporal “wumpus physics”
               t, a counter, initially 0, indicating time
               plan, an action sequence, initially empty

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  TELL the KB the temporal “physics” sentences for time t
  safe  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \text{OK}_{x,y}^t) = \text{true}\}$ 
  if ASK(KB, Glittert) = true then
    plan  $\leftarrow [\text{Grab}] + \text{PLAN-ROUTE}(\text{current}, \{[1,1]\}, \text{safe}) + [\text{Climb}]$ 
  if plan is empty then
    unvisited  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, L_{x,y}^{t'}) = \text{false} \text{ for all } t' \leq t\}$ 
    plan  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \text{unvisited} \cap \text{safe}, \text{safe})$ 
  if plan is empty and ASK(KB, HaveArrowt) = true then
    possible_wumpus  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg W_{x,y}) = \text{false}\}$ 
    plan  $\leftarrow \text{PLAN-SHOT}(\text{current}, \text{possible\_wumpus}, \text{safe})$ 
  if plan is empty then // no choice but to take a risk
    not_unsafe  $\leftarrow \{[x, y] : \text{ASK}(\text{KB}, \neg \text{OK}_{x,y}^t) = \text{false}\}$ 
    plan  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \text{unvisited} \cap \text{not\_unsafe}, \text{safe})$ 
  if plan is empty then
    plan  $\leftarrow \text{PLAN-ROUTE}(\text{current}, \{[1, 1]\}, \text{safe}) + [\text{Climb}]$ 
  action  $\leftarrow \text{POP}(\text{plan})$ 
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t  $\leftarrow t + 1$ 
  return action

```

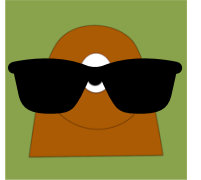
The Wumpus Agent



```
function PLAN-ROUTE(current, goals, 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)
```

The Wumpus Agent



We need more!

Effect axioms

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

We need extra axioms about the world.

Frame problem! – representational & inferential

Frame axioms:

$$\begin{aligned} Forward^t &\Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1}) \\ Forward^t &\Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1}) \end{aligned}$$

Successor-state axioms:

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

Expressiveness limitation of propositional logic

- KB contains "physics" sentences for every single square.
- For every time t and every location $[x,y]$,

$$L^t_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L^{t+1}_{x+1,y}$$

- Rapid proliferation of clauses!

Why?

- Fundamentals behind *SAT/SMT solvers*.
- Highly specialised and optimised tools.
 - Capable of solving problems with thousands of propositions and millions of constraints, despite NP-completeness and exponential algorithms!
- Close relation to CSPs and *optimization* problems.
- Very large array of applications, e.g.:
 - Circuit routing and testing, automatic test generation, formal verification, planning & scheduling, configuration/customisation, etc.