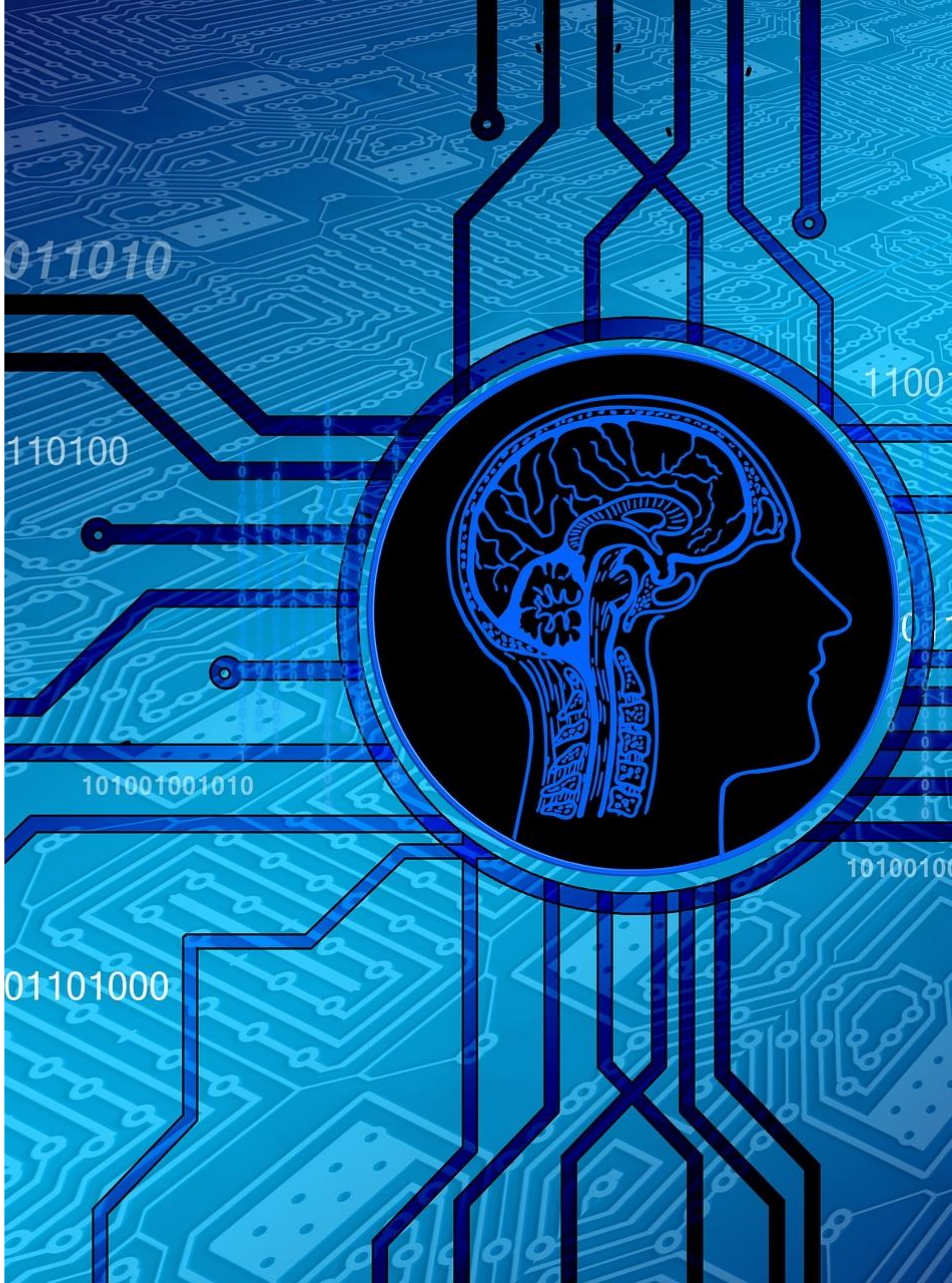
# Effective Propositional Inference

Informatics 2D: Reasoning and Agents

**Lecture 9**

*Adapted from slides provided by Dr Petros Papapanagiotou*



# 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<br>• Conjunction of clauses |
| Clause | • Disjunction of literals |
| Literal | • Proposition symbol or negated proposition symbol |

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

# Conversion to CNF

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

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

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

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

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

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

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

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

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

# 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

DPLL

# 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 $(A \lor \neg B)$, $(\neg B \lor \neg C)$, $(C \lor A)$:
    - 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).

DPLL

**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* – $P$, *model* ∪ {$P$=*value*})
  $P$, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)
  **if** $P$ is non-null **then return** DPLL(*clauses*, *symbols* – $P$, *model* ∪ {$P$=*value*})
  $P$ ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  **return** DPLL(*clauses*, *rest*, *model* ∪ {$P$=*true*}) **or**
        DPLL(*clauses*, *rest*, *model* ∪ {$P$=*false*})

# The DPLL algorithm

# Tautology Deletion (Optional)

- **Tautology**: both a proposition and its negation in a clause.
  - e.g. (A, B, ¬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:

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

- 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**

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

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

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

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

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

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

# 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**

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

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

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

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

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

**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**

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

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

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

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

**T**

# Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

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

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**

**T**

**T**

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

**T**

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

**T**

# Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

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

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**

**T**

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

**T**

**T**

**T**

# Solution

Pure symbol heuristic:

- No literal is pure.

Unit clause heuristic:

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

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**

**T**

**T**

**T**

**T**

**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* ← 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*

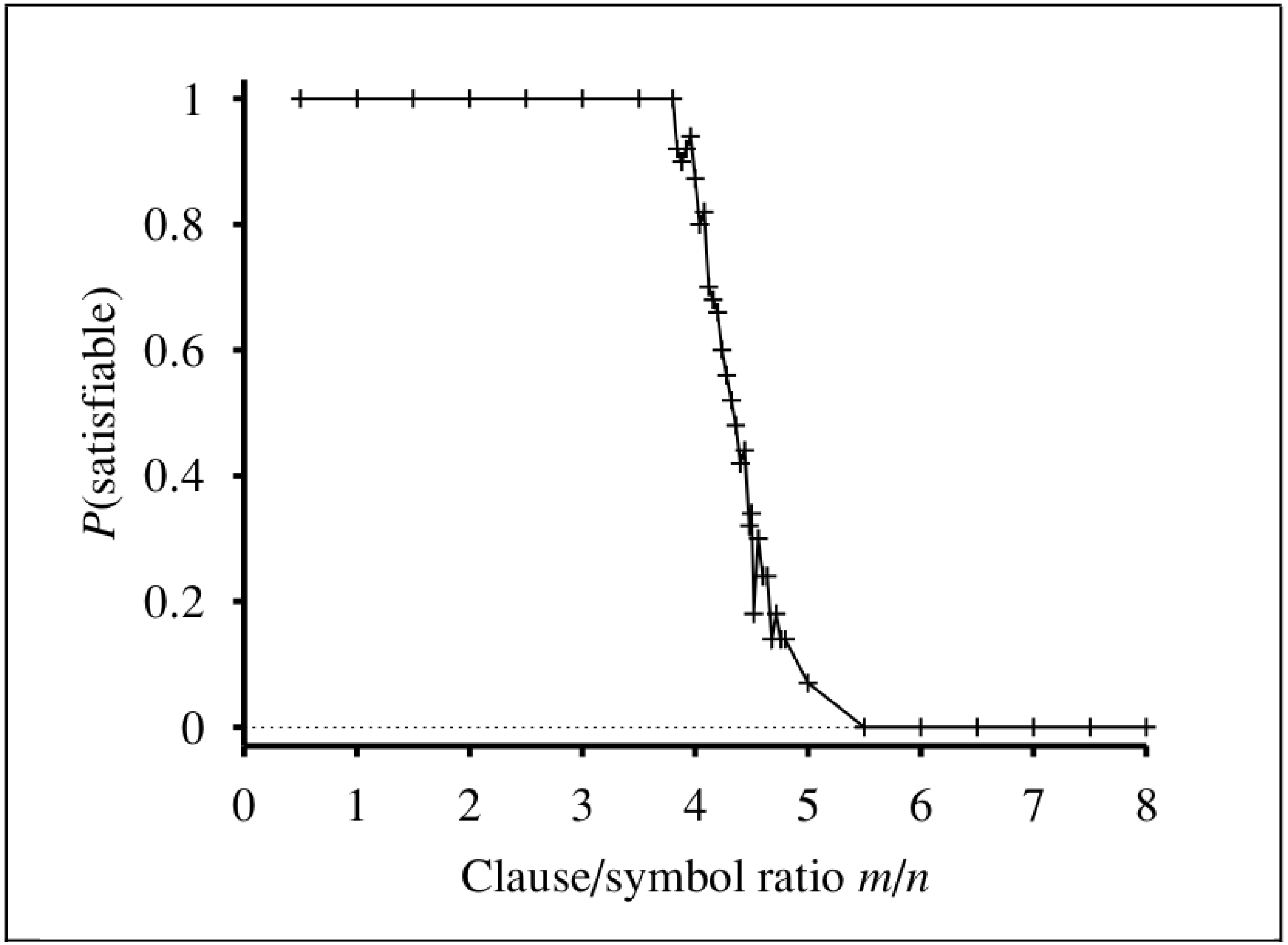
# The WalkSAT algorithm

# Hard satisfiability problems

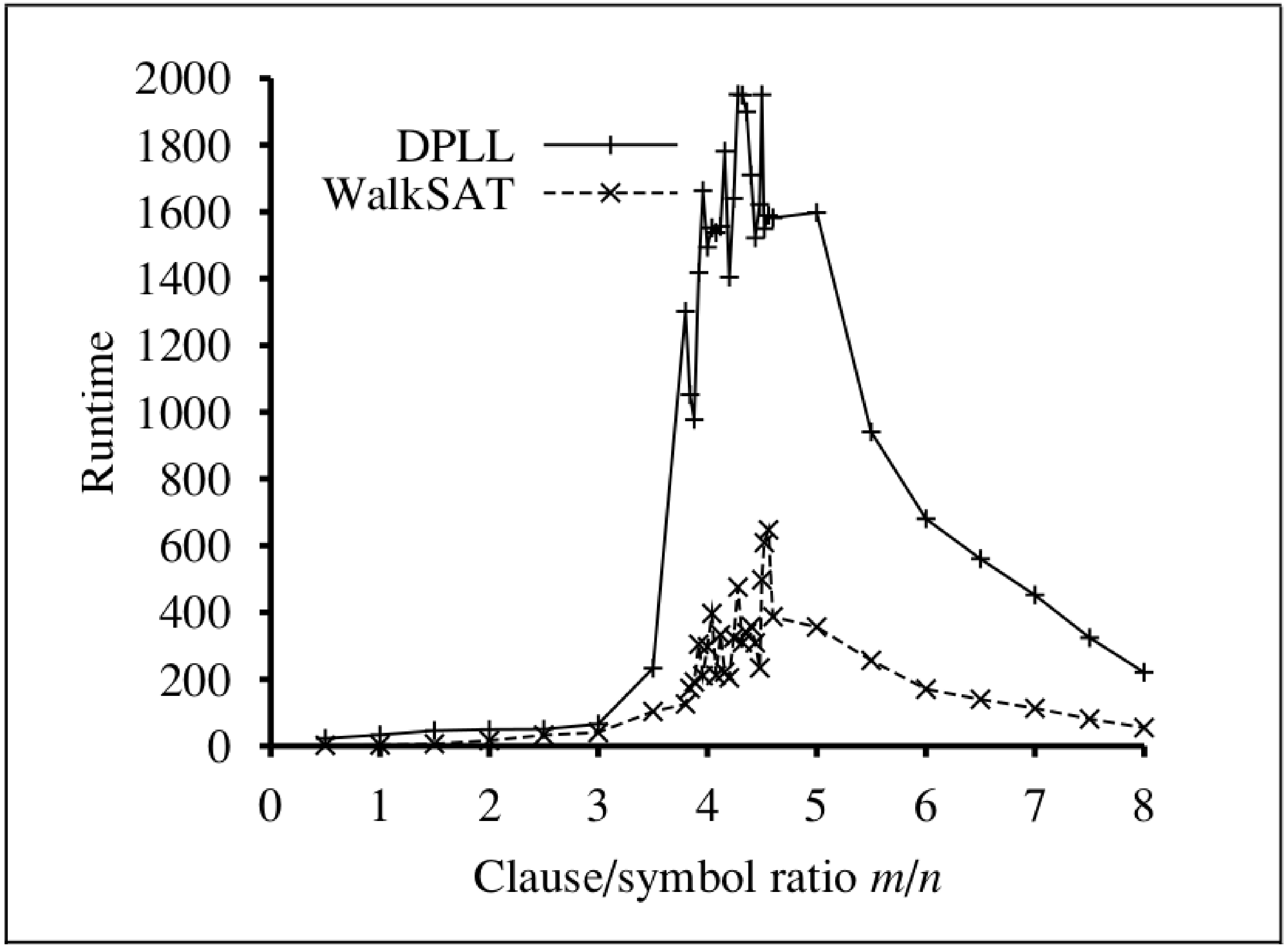
- Consider random 3-CNF sentences.
  - Example:

$$(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C)$$

  - $m$ = number of clauses
  - $n$ = number of symbols
- Hard problems seem to cluster near $m/n = 4.3$ (critical point)

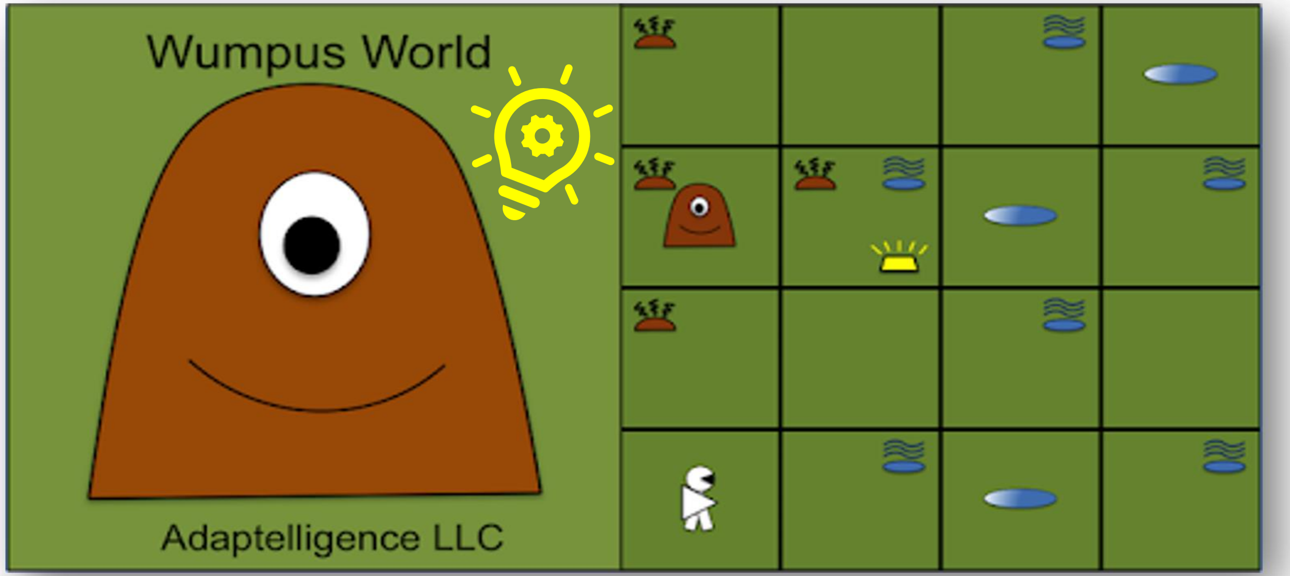



# Hard satisfiability problems




# 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:
  - $\neg P_{1,1}$
  - $\neg W_{1,1}$
  - $B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$
  - $S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$
  - $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$
  - $\neg W_{1,1} \vee \neg W_{1,2}$
  - $\neg W_{1,1} \vee \neg W_{1,3}$
  - …
- 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 $t$
  $safe \leftarrow \{[x, y] : \text{ASK}(KB, OK^t_{x,y}) = true\}$
  **if** $\text{ASK}(KB, Glitter^t) = true$ **then**
    $plan \leftarrow [Grab] + \text{PLAN-ROUTE}(current, \{[1,1]\}, safe) + [Climb]$
  **if** *plan* is empty **then**
    $unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^{t'}_{x,y}) = false$ for all $t' \leq t\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap safe, safe)$
  **if** *plan* is empty and $\text{ASK}(KB, HaveArrow^t) = true$ **then**
    $possible\_wumpus \leftarrow \{[x, y] : \text{ASK}(KB, \neg W_{x,y}) = false\}$
    $plan \leftarrow \text{PLAN-SHOT}(current, possible\_wumpus, safe)$
  **if** *plan* is empty **then** // no choice but to take a risk
    $not\_unsafe \leftarrow \{[x, y] : \text{ASK}(KB, \neg OK^t_{x,y}) = false\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap not\_unsafe, safe)$
  **if** *plan* is empty **then**
    $plan \leftarrow \text{PLAN-ROUTE}(current, \{[1, 1]\}, safe) + [Climb]$
  $action \leftarrow \text{POP}(plan)$
  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* ← 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:

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$

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],

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

- 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.