8

Effective Propositional Inference

Claudia Chirita

School of Informatics, University of Edinburgh

Based on slides by: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann, Vaishak Belle

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

8.a

DPLL algorithm

CLAUSAL FORM · CNF

Both DPLL and WalkSAT algorithms 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. $(A, \neg B), (B, \neg C)$ representing $(A \vee \neg B) \wedge (B \vee \neg C)$

CONVERSION TO CNF · EXAMPLE

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

Eliminate $\leftrightarrow$ replacing $\alpha \leftrightarrow \beta$ by $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
$(B_{1,1} \rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \rightarrow B_{1,1})$

Eliminate $\rightarrow$ replacing $\alpha \rightarrow \beta$ by $\neg\alpha \vee \beta$
$(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

Move $\neg$ inwards using de Morgan's rules
$(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
possibly also eliminating double-negation: replacing $\neg(\neg\alpha)$ by α

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 ALGORITHM

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

Improvements over truth table enumeration:

- early termination
- pure symbol heuristic
- unit clause heuristic

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

PURE SYMBOL HEURISTIC

Pure symbol = appears with the same “sign” or 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. (for satisfiability) let A and $\neg B$ both be true

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 ALGORITHM

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*}))

TAUTOLOGY DELETION

Tautology = both a proposition symbol 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.

QUESTION TIME!

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 these heuristics.

QUESTION TIME!

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

Symbols $S_{1,1}, S_{1,2}, S_{2,1}, W_{2,2}$

Pure symbol heuristic No literal is pure.

Unit clause heuristic $S_{2,1}$ is true; $S_{1,1}$ and $S_{1,2}$ are false.

Early termination $(\neg S_{1,1}, W_{2,2}), (\neg S_{1,2}, W_{2,2})$ are both true.
$(\neg W_{2,2}, S_{2,1}, S_{1,1}, S_{1,2})$ is true.

Unit clause heuristic $\neg S_{2,1}$ is false, so $(\neg S_{2,1}, W_{2,2})$ becomes unit clause.
$W_{2,2}$ must be true.

Exploratory explanation of DPLL

8.b

WalkSAT algorithm

WALKSAT ALGORITHM

Incomplete, local search algorithm

Evaluation function	the min-conflict heuristic of minimizing the number of unsatisfied clauses

Balance between greediness and randomness

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, 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
```

Checks for satisfiability by randomly flipping the values of variables.

HARD SATISFIABILITY PROBLEMS

Consider random 3-CNF sentences: **3SAT problem**

EXAMPLE

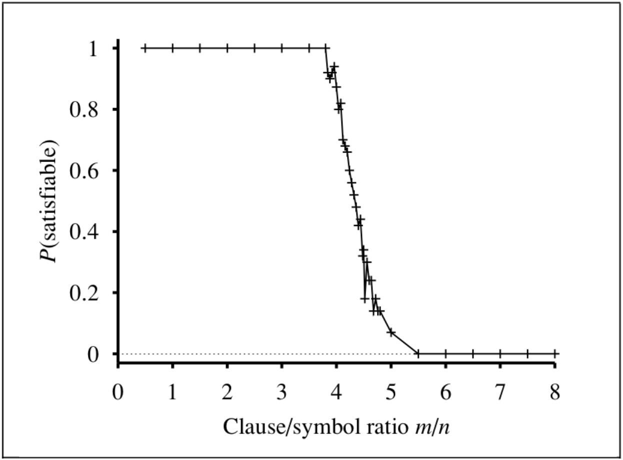
$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E)$$
$$\wedge\, (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

m number of clauses

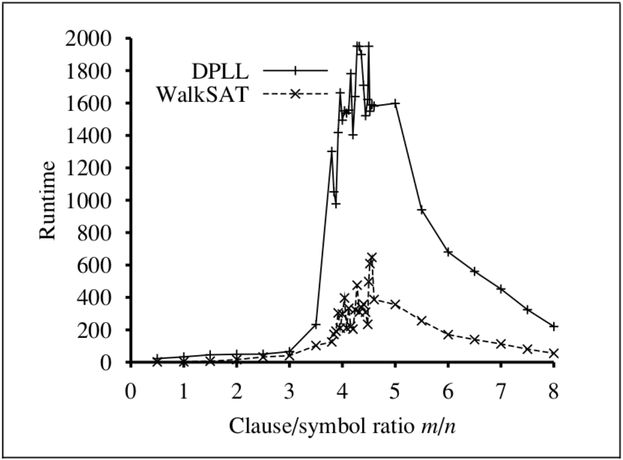
n number of symbols

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, $n = 50$

8.c

Inference in the Wumpus World

A wumpus-world agent using propositional logic

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

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

WUMPUS WORLD AGENT

```
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 ← {[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 is empty then
    unvisited ← {[x, y] : ASK(KB, L^{t'}_{x,y}) = false for all t' ≤ t}
    plan ← PLAN-ROUTE(current, unvisited ∩ safe, safe)
  if plan is empty 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 is empty then   // no choice but to take a risk
    not_unsafe ← {[x, y] : ASK(KB, ¬ OK^t_{x,y}) = false}
    plan ← PLAN-ROUTE(current, unvisited ∩ not_unsafe, safe)
  if plan is empty then
    plan ← PLAN-ROUTE(current, {[1, 1]}, safe) + [Climb]
  action ← POP(plan)
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

WUMPUS WORLD 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($problem$)

WE NEED MORE!

EFFECT AXIOMS

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

We need extra axioms about the world.

FRAME PROBLEM

Frame axioms

$$\text{Forward}^t \rightarrow (\text{HaveArrow}^t \leftrightarrow \text{HaveArrow}^{t+1})$$
$$\text{Forward}^t \rightarrow (\text{WumpusAlive}^t \leftrightarrow \text{WumpusAlive}^{t+1})$$

Successor-state axioms

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

EXPRESSIVENESS LIMITATION OF PL

The KB contains “physics” sentences for every single square.

For every time t and every location $[x, y]$

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

Rapid proliferation of clauses.