

2-SAT Resolution

Samy Aittahar

ULiege - INFO0049

Tuesday 13th March, 2018

Definitions

Beforehand, we consider the following definitions :

- ▶ An atom (e.g. x_0) is a simple proposition ;
- ▶ A literal is an atom or its negation (e.g. $x_0, \neg x_0$) ;
- ▶ X is a finite set of atoms and Y is the finite set of literals defined from X (i.e. $\forall x \in X, x \in Y \wedge \neg x \in Y$) ;
- ▶ The lexicon of a formula Φ is the set of atoms $X_\Phi \subseteq X$ with at least one occurrence in Φ ;
- ▶ A clause is a disjunction of a finite number of literals ;

Definitions (cont'd)

- ▶ A formula in CNF (Conjunctive Normal Form) is a conjunction of a finite number of clauses ;
- ▶ A valuation $V : X \rightarrow \{0, 1\}$ is a function which assigns a truth value to each atom $x \in X$;
- ▶ A valuation defined on a lexicon naturally extends to all formulas defined on such a lexicon; if Φ is such a formula, then $v(\Phi)$ is the associated truth value ;
- ▶ A model of a formula is a valuation that satisfies the formula (i.e. the valuation of the formula is 1) ;
- ▶ A formula is consistent if it admits at least one model.

CNF-SAT : Problem statement

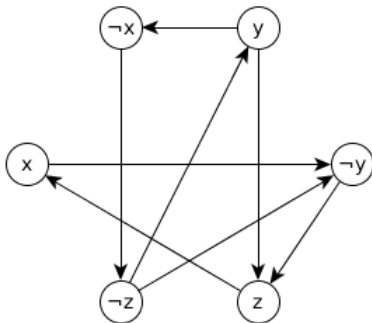
Goal : Given a CNF formula Φ defined on the lexicon X_Φ , give a model if Φ is consistent.

CNF-SAT : Properties and resolution

- ▶ NP Complete problem. No general polynomial-time solution is known.
- ▶ However, with at most 2 literals per clause (2-SAT form), polynomial-time solving algorithms are known.
- ▶ Steps (exactly two literals per clause) :
 - ▶ Build the conditional graph from the 2-SAT formula $(A \vee B \Leftrightarrow (\neg A \Rightarrow B) \vee (\neg B \Rightarrow A))$;
 - ▶ Fetch subgraphs that are strongly connected, i.e. each vertex is reachable from the others.
 - ▶ Decision rule : if a literal and its negation are involved in the same component, formula is not consistent.
 - ▶ If the formula is consistent, assign 1 to literals that are not negations and 0 to the others.

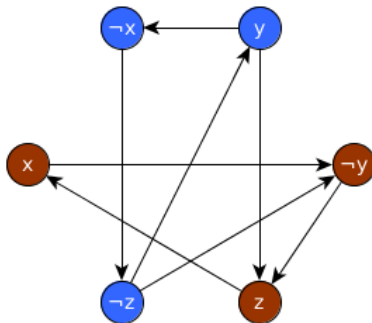
Implication graph example

Implication graph built from the following formula
 $(\neg x \vee \neg y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$



Strongly connected component (SCC)

Implication graph built from the following formula
 $(\neg x \vee \neg y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$



Model : $x \rightarrow 1, y \rightarrow 0, z \rightarrow 1$

Known algorithms to fetch SCCs

- ▶ Kosaraju's algorithm. Two depths searches in respectively the graph and its transpose.
 - ▶ Transpose of a graph G is G with reverted edges.
- ▶ Tarjan's algorithm. One depth search with a stack. Use of indexes assigned to vertices to detect root nodes and then build SCCs.

A more Prolog-friendly algorithm ?

Path-based strong component algorithm

- ▶ Last (known) version of this algorithm by [Gabow,2000].
- ▶ Two stack are used. One to keep track of the current component and the second to keep track of the search path.
- ▶ List insertion/read in Prolog is done in a stack fashion.

Path-based strong component algorithm

Algorithm Strongly connected components with stacks and DFS

```
1:  $C \leftarrow 0, S \leftarrow [], P \leftarrow []$ 
2:  $Prs \leftarrow \{\}, Sccs \leftarrow \{\}$ 
3: procedure PBCSA( $v, E$ )
4:   Prepend  $v$  to  $P$  and  $S$ 
5:    $Prs \leftarrow Prs \cup \{v, C\}$ 
6:    $C \leftarrow C + 1$ 
7:   for  $w$  in  $neighbors(v, E)$  do
8:     if  $\{w, N\} \notin Prs$  then
9:       PBCSA( $w, E$ )
10:    else if  $\nexists scc \in Sccs | w \in scc$  then
11:       $M \leftarrow +\infty$ 
12:      while  $M > N$  do
13:         $z \leftarrow Pop(P)$ 
14:        Pick  $\{z, m\} \in Prs$ 
15:         $M \leftarrow m$ 
16:      end while
17:    end if
18:  end for
19:  if  $Top(P) = v$  then
20:     $scc \leftarrow \{\}$ 
21:     $lastpop \leftarrow Pop(S)$ 
22:    while  $lastpop \neq v$  do
23:       $scc \leftarrow scc \cup lastpop$ 
24:       $lastpop \leftarrow Pop(S)$ 
25:    end while
26:     $Pop(P)$ 
27:  end if
28:  return  $I$ 
29: end procedure
30: Given a graph  $\{V, E\}$  :
31: for  $v$  in  $V$  do
32:   if  $\{v, \} \notin Prs$  then
33:     PBCSA( $v, E$ )
34:   end if
35: end for
```

Path-based strong component algorithm - Example

Execute the algorithm through the example shown in Slide 6.

2SAT - Scheduling problem

- ▶ We consider a classroom with n teachers and m cohorts of students.
- ▶ Each teacher have a set of working hours slots in which they are available in a week.
- ▶ Each teacher have a fixed number of hours to spend with each cohort of students.
- ▶ A teacher cannot be assigned to two cohorts of students at the same hour slot.
- ▶ A cohort of students cannot be assigned to two teachers at the same hour slot.
- ▶ The goal of the scheduling problem is to assign hours slots to teachers and cohorts in order to fulfill the time each teacher has to spend with cohorts of students.
 - ▶ NP-Complete. We consider here that each teacher has only one or two hours to spend with the students.

SAT - Homework

- ▶ Implement a 2SAT Solver using at least two algorithms from the literature (including the presented one).
 - ▶ Make sure to separate and explain clearly each algorithm you have implemented in your source code file.
- ▶ Benchmark them with a common set of formulas, providing examples and time execution.
- ▶ Provide a formalisation of the scheduling problem.
- ▶ Build instances of the scheduling problem and solve them using your 2SAT solver. Provide examples and time execution.

Homework - Input/Output (2SAT Solver)

- ▶ The input is a list of pairs of literals (an atom or a term $n(A)$ where A is an atom).
- ▶ The output is either :
 - ▶ The empty list if the formula Φ described in the input is not consistent.
 - ▶ A list of pairs in which each atom in the lexicon X_Φ is associated to a $\{0, 1\}$ valuation.

Homework - Input/Output (Scheduling Solver)

- ▶ The inputs are :
 - ▶ A list of list of natural numbers (only 1 or 2) R of size $n \times m$.
 - ▶ A list of list of list of binary numbers A of size $n \times 5 \times 10$.
- ▶ The output is either
 - ▶ An empty list if there is no solution,
 - ▶ The number of days + a list of quadruplets that contains for each assignment the teacher i , the cohort j , the day d and the hour slot i .

That's all folks

Start with *<https://en.wikipedia.org/wiki/2-satisfiability>* to know more about 2SAT