Theory of Hybrid Systems Informatik 2
RWTH AACHEN UNIVERSITY

Satisfiability Checking - WS 2016/2017

Series 3

gereon.kremer@cs.rwth-aachen.de
https://ths.rwth-aachen.de/teaching/

Exercise 1

Consider the following formula:

$$\begin{array}{llll} c_1 : (\neg b \vee c \vee \neg d) & \wedge & c_2 : (\neg b \vee d) & \wedge \; c_3 : (\neg c \vee \neg d) \quad \wedge \\ c_4 : (\neg a \vee c) & \wedge & c_5 : (a \vee b) & \wedge \; c_6 : (\neg a \vee \neg c) \quad \wedge \\ c_7 : (a \vee \neg c \vee \neg d) & & & \end{array}$$

a) Give valid initial watch lists for each literal occurring in the formula. Note that there are several possibilities to do so.

b) Draw the implication graph for the initial decision using the variable state independent decaying sum (VSIDS) decision heuristic In the case that the activities of the literals are the same, choose the alphabetically smallest variable first (a<b<c<d) and assign it by default to false. Is there a conflict? If so, apply conflict resolution to the conflict which occurred, i.e., use resolution on the conflict clause and the clauses that implied the current assignments, in reverse order back until the first unique implication point is reached. Furthermore, answer the following questions:

   i) Which asserting clause is generated?
   ii) Which assignments would be undone using conflict-driven non-chronological backtracking?
   iii) What does the DPLL algorithm do after backtracking?

c) Give the watch lists resulting from the initial decision, and also the lists resulting from the propagation of the first implication drawn in part b).

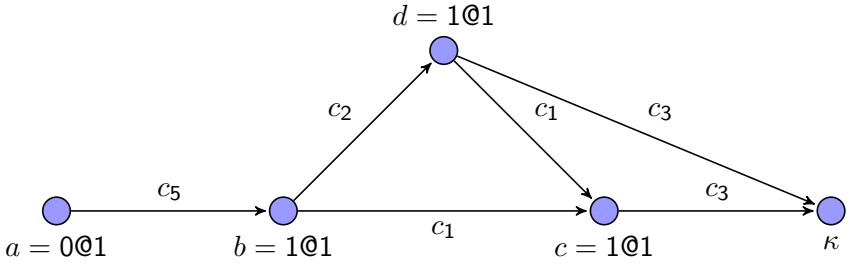
d) Is the formula satisfiable? If so, give a satisfying assignment. Otherwise, give an unsatisfiable core, i.e., a preferably small subset of the original set of clauses which is unsatisfiable.

Solution:

a) Possible initial watch lists for the given formula:

$$\begin{array}{llllllll} a\text{:} & \{c_5,\, c_7\} & b\text{:} & \{c_5\} & c\text{:} & \{c_1,\, c_4\} & d\text{:} & \{c_2\} \\ \neg a\text{:} & \{c_4,\, c_6\} & \neg b\text{:} & \{c_1,\, c_2\} & \neg c\text{:} & \{c_3,\, c_6,\, c_7\} & \neg d\text{:} & \{c_3\} \end{array}$$

b) Implication graph for $a = 0@1$:



- Conflicting clause: *cl* $= c_3 : (\neg c \vee \neg d)$. Latest assigned literal in *cl* is $\neg c$. Variable of $\neg c$ is c. Antecedent of c is c_1. Resolution of *cl* and c_1 for c:

$$\frac{\textit{cl} : (\neg \mathbf{c} \vee \neg d) \quad c_1 : (\neg b \vee \mathbf{c} \vee \neg d)}{c_8 : (\neg b \vee \neg d)}$$

- *cl* $= c_8 : (\neg b \vee \neg d)$. Latest assigned literal in *cl* is $\neg d$. Variable of $\neg d$ is d. Antecedent of d is c_2. Resolution of *cl* and c_2 for d:

$$\frac{\textit{cl} : (\neg b \vee \neg \mathbf{d}) \quad c_2 : (\neg b \vee \mathbf{d})}{c_9 : (\neg b)}$$

- The clause c_9 contains exactly one variable that has been assigned in the current decision level and is, hence, asserting (UIP), so we can stop resolution here.

i) c_9 is the result of the conflict resolution and thus, an asserting clause.

ii) All assignments (at decision level 1) get undone.

iii) The solver will assign $b = 0@0$ and propagate this assignment at decision level 0.

c) According to the watch lists of part b) only the clauses for a have to be considered.

- Propagation of $a = 0@1$ in c_5: implication of assignment $b = 1@1$.
- Propagation of $a = 0@1$ in c_7: move c_7 from the watch list of a to the list of $\neg d$.

The updated watch lists are:

a:	$\{c_5\}$	b:	$\{c_5\}$	c:	$\{c_1,\ c_4\}$	d:	$\{c_2\}$
$\neg a$:	$\{c_4,\ c_6\}$	$\neg b$:	$\{c_1,\ c_2\}$	$\neg c$:	$\{c_3,\ c_6,\ c_7\}$	$\neg d$:	$\{c_3,\ c_7\}$

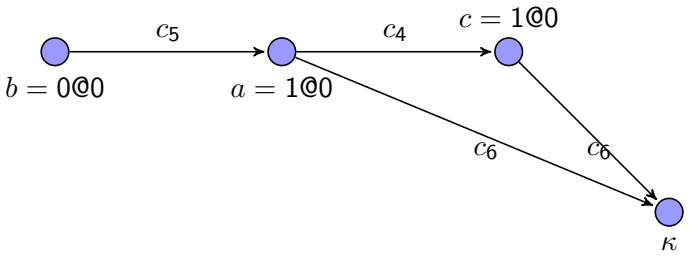
The only implication $b = 1@1$ is propagated in the appropriate clauses as follows.

- Propagation of $b = 1@1$ in c_1: move c_1 from the watch list of $\neg b$ to the list of $\neg d$.
- Propagation of $b = 1@1$ in c_2: assignment $d = 1@1$.

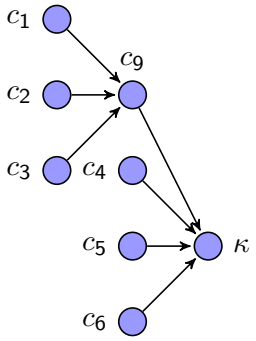
The updated watch lists are:

a:	$\{c_5\}$	b:	$\{c_5\}$	c:	$\{c_1,\ c_4\}$	d:	$\{c_2\}$
$\neg a$:	$\{c_4,\ c_6\}$	$\neg b$:	$\{c_2\}$	$\neg c$:	$\{c_3,\ c_6,\ c_7\}$	$\neg d$:	$\{c_1,\ c_3,\ c_7\}$

d) The formula is unsatisfiable. When the procedure propagates $b = 0@0$, we get the following implication graph:



Thus the resolution graph for the whole search procedure look as follows:



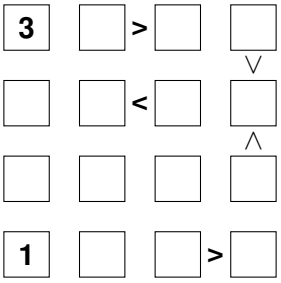
Unsatisfiable core: $\{c_1, c_2, c_3, c_4, c_5, c_6\}$.

Exercise 2

Consider the game **Unequal** having the following rules:
You have an $n \times n$ square grid; each square may contain a number from 1 to n, and some squares have clue signs between them. Your aim is to fully populate the grid with numbers such that

- each row contains only one occurrence of each number,
- each column contains only one occurrence of each number,
- and all the clue signs are satisfied.

Abbildung 1: An example starting instance of Unequal.



a) Formulate the rules of Unequal for an arbitrary square grid in propositional logic. (With no clue signs between the grids and no square fixed to a value as in the example.)

b) Formulate the starting instance given by Figure 1 in propositional logic such that the resulting formula is satisfiable iff this instance has a solution according to the game's rules.

c) If the formula is not yet in CNF, convert it into CNF.

d) Transfer the formula into the standard SAT input format (DIMACS) and let MiniSat solve it. Is the formula satisfiable?

Solution:

a) Variables:

$$g(i,\ j,\ k) \quad := \quad \bigwedge_{\substack{r\ \in\ PosBits(i)\\ s\ \in\ PosBits(j)\\ t\ \in\ PosBits(k)}} b^t_{r,s} \ \wedge \ \bigwedge_{\substack{r\ \in\ NegBits(i)\\ s\ \in\ NegBits(j)\\ t\ \in\ NegBits(k)}} \neg b^t_{r,s}$$

$$\hat{=} \quad \text{grid at the coordinates } (i,\ j) \text{ has the number } k,$$

$$\text{where } i,\ j,\ k\ \in\ N := \{1, \ldots, n\}$$

$$PosBits(k) \quad := \quad \{\ i \mid k = \sum_{j=0}^{\lfloor log(n) \rfloor} z_j \cdot 2^j,\ \forall j : z_j \in \{0,\ 1\},\ z_i = 1\}$$

$$NegBits(k) \quad := \quad \{\ i \mid k = \sum_{j=0}^{\lfloor log(n) \rfloor} z_j \cdot 2^j,\ \forall j : z_j \in \{0,\ 1\},\ z_i = 0\}$$

Hence the only propositional variables we declare are the

$$b^t_{r,s}, \text{ where } r,\ s,\ t \in \{1,\ \ldots,\ \lfloor log(n) \rfloor\}$$

and $g(i,\ j,\ k)$ is used as short form. The number of variables we introduce is $\lfloor log(n) \rfloor^3$.

Game rules:

- Each grid has at least one number:

$$\varphi_{grids} := \bigwedge_{i,\ j\ \in\ N} \left(\bigvee_{k \in N} g(i,\ j,\ k) \right)$$

  It would not be necessary, if $n = 2^m$.
- Each pair of grids within a row has different numbers:

$$\varphi_{rows} := \bigwedge_{\substack{i,\ j,\ k,\ l\ \in\ N\\ l\ >\ j}} (\neg g(i,\ j,\ k) \vee \neg g(i,\ l,\ k))$$

- Each pair of grids within a column has different numbers:

$$\varphi_{columns} := \bigwedge_{\substack{i,\ j,\ k,\ l\ \in\ N\\ l\ >\ i}} (\neg g(i,\ j,\ k) \vee \neg g(l,\ j,\ k))$$

We do not need to state that a grid has not more than one number, because it is implied by φ_{rows} and $\varphi_{columns}$, respectively.

b) First, we formulate that the grid at coordinates $(r,\ s)$ is greater than the grid at coordinates $(t,\ u)$.

$$\varphi_{>}(r,\ s,\ t,\ u) := \bigwedge_{\substack{k, l\ \in\ N\\ l\ >\ k}} (\neg g(r,\ s,\ k) \vee \neg g(t,\ u,\ l))$$

We do not need to cover the case that $l = k$, as φ_{rows} and $\varphi_{columns}$ already exclude this case. Then, the starting instance given by Figure 1 has a solution according to the game's rules, if the following formula is satisfiable.

$$\begin{aligned} \varphi \quad := \quad & \varphi_{grid} \ \wedge \ \varphi_{rows} \ \wedge \ \varphi_{columns} \\ \wedge \quad & \varphi_{>}(1,\ 2,\ 1,\ 3) \ \wedge \ \varphi_{>}(2,\ 3,\ 2,\ 2) \ \wedge \ \varphi_{>}(1,\ 4,\ 2,\ 4) \\ \wedge \quad & \varphi_{>}(3,\ 4,\ 2,\ 4) \ \wedge \ \varphi_{>}(4,\ 3,\ 4,\ 4) \ \wedge \ g(1,\ 1,\ 3) \ \wedge \ g(4,\ 1,\ 1) \end{aligned}$$

c) The $g(i,\ j,\ k)$ are actually conjunctions of literals and appear in disjunctions, therefore φ is not yet in CNF. To achieve CNF we replace each $g(i,\ j,\ k)$ by a fresh propositional variable $g^k_{i,j}$ and append the formula

$$\varphi_{g(i,\ j,\ k)} := \bigwedge_{\substack{r \in PosBits(i) \\ s \in PosBits(j) \\ t \in PosBits(k)}} (\neg g^k_{i,j} \vee b^t_{r,s}) \ \wedge \bigwedge_{\substack{r \in NegBits(i) \\ s \in NegBits(j) \\ t \in NegBits(k)}} (\neg g^k_{i,j} \vee \neg b^t_{r,s})$$

d) Output of MiniSat 2.0 beta:

```
============================[ Problem Statistics ]=============================
|                                                                             |
|  Number of variables:   64                                                  |
|  Number of clauses:     240                                                 |
|  Parsing time:          0.00 s                                              |
============================[ Search Statistics ]==============================
| Conflicts |           ORIGINAL         |         LEARNT          | Progress |
|           |    Vars   Clauses Literals |   Limit  Clauses Lit/Cl |          |
===============================================================================
|         0 |      70       170      368 |      56        0   -nan |  0.000 % |
===============================================================================
Verified 170 original clauses.
Assignments:
restarts            : 1
conflicts           : 1              (250 /sec)
decisions           : 29             (0.00 % random) (7250 /sec)
propagations        : 93             (23250 /sec)
conflict literals   : 3              (0.00 % deleted)
Memory used         : 14.74 MB
CPU time            : 0.004 s

SATISFIABLE
```