Exercises for the Lecture Practical SAT Solving	Summer Semester 2019	Assignment 3

Institute for Theoretical Informatics, Karlsruhe Institute of Technology (KIT) 6. 6.2019

Tomas Balyo, Carsten Sinz Due 20. 6.2019

Exercise 1: (DPLL) [5 Points] Simulate modern DPLL (from Slide 16 of Lecture 5 slides) by hand on the formula $F = (x_3 \vee x_4 \vee \bar{x}_1 \vee x_5) \wedge (\bar{x}_3 \vee x_4 \vee x_5) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_1) \wedge (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_5) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee x_5)$. Select branching literals in the order $x_1, x_2, x_3, \ldots$.

Exercise 2: (Stålmarck's Method) [5 Points] Simulate by hand Stålmarcks Method (from Slide 8 of Lecture 6 slides) by hand on the formula from exercise 1. Select variables in the order $x_1, x_2, x_3, \ldots$.

Exercise 3: (CDCL) [7 Points] Simulate CDCL (from slide 24 of Lecture 7) by hand on the formula $F = (x_1 \vee x_{13}) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_{14}) \wedge (x_3 \vee x_{15}) \wedge (x_4 \vee x_{16}) \wedge (\bar{x}_5 \vee \bar{x}_3 \vee x_6) \wedge (\bar{x}_5 \vee \bar{x}_7) \wedge (\bar{x}_6 \vee x_7 \vee x_8) \wedge (\bar{x}_4 \vee \bar{x}_8 \vee \bar{x}_9) \wedge (\bar{x}_1 \vee x_9 \vee \bar{x}_{10}) \wedge (x_9 \vee x_{11} \vee \bar{x}_{14}) \wedge (x_{10} \vee \bar{x}_{11} \vee x_{12}) \wedge (\bar{x}_2 \vee \bar{x}_{11} \vee \bar{x}_{12})$ Select branching literals in the order $x_1, x_2, x_3, \ldots$ Draw the implication graph for each conflict and learn the 1-UIP clause.

Exercise 4: (Local Search Challenge) [10(+10) points Points] Implement a (stochastic) local search SAT solver. Follow the SAT Competition input/output format `http://www.satcompetition.org/2004/format-solvers2004.html` For a working solver you get 10 points. The author of the best solver receives a bonus of 10 points. The solvers will be evaluated on satisfiable random 3-SAT problems. (like the ones here: `https://baldur.iti.kit.edu/sat/files/local-sat.zip`). You don't need to start from scratch, use solver stub from the `local-sat.zip` package, it already contains the input parsing.

Exercise 5: (Hidoku Challenge) [12(+12) Points] Hidoku a.k.a Hidato a.k.a Number Snake is a logic puzzle where the goal is to fill a grid with consecutive numbers that connect horizontally, vertically, or diagonally. The grid is rectangular and some of the cells are pre-filled. Example:

1			5
	7		
			14
		16	

Unsolved Hidoku

1	3	4	5
2	7	6	13
8	11	12	14
9	10	16	15

It's solution

1			5
2			14
		16	

Unsatisfiable Hidoku

The input is a single string looking like this (for the above example):
`4,4:1,0,0,5;0,7,0,0;0,0,0,14;0,0,16,0;`
The first two numbers are the width and heigth of the grid followed by the values separated by commas, rows are separated by semicolons, 0 represents an empty cell. The output format:
`sol:1,3,4,5;2,7,6,13;8,11,12,14;9,10,16,15;`
A Hidoku puzzle may be unsatisfiable, in that case ouput `sol:UNSAT`

Implement a SAT solving based Hidoku solver. For a working solver you get 12 points. The fastest solver will receive a bonus of 12 points. Here are some example inputs `https://baldur.iti.kit.edu/sat/files/hidokus.txt` for testing.