SAT and SMT Solving — SS 2018 — LVA 703048

Exercises 6 — May 3, 2018

[3] **1** Consider the following set of equations E:

$$\mathsf{a} \approx \mathsf{b} \qquad \mathsf{f}(\mathsf{a}) \approx \mathsf{b} \qquad \mathsf{f}(\mathsf{b}) \approx \mathsf{c}$$
$$\mathsf{g}(\mathsf{a}) \approx \mathsf{g}(\mathsf{g}(\mathsf{b})) \qquad \mathsf{f}(\mathsf{a}) \approx \mathsf{g}(\mathsf{b}) \qquad \mathsf{f}(\mathsf{c}) \approx \mathsf{f}(\mathsf{g}(\mathsf{c}))$$

and use congruence closure to determine whether the following hold:
(a) $E \vDash \mathsf{a} \approx \mathsf{c}$ (b) $E \vDash \mathsf{f}(\mathsf{c}) \approx \mathsf{c}$

[3] **2** Determine satisfiability of the following formula:

$$(\mathsf{g}(x_3) \neq \mathsf{g}(x_4) \vee \mathsf{g}(x_1) = x_1) \wedge (x_3 = x_4 \vee x_1 = x_2) \wedge (\mathsf{f}(x_1, x_3) \neq x_1 \rightarrow x_1 = \mathsf{f}(x_1, x_4)) \wedge$$
$$x_1 \neq x_2 \wedge (\mathsf{g}(x_3) \neq \mathsf{g}(x_4) \vee \mathsf{g}(\mathsf{f}(x_1, x_3)) \neq \mathsf{g}(x_1))$$

by transforming the formula to CNF, applying DPLL(T), and using congruence closure to check T-consistency of models.

[2] ⋆**3** In a 3×3 magic square the numbers $1-9$ are arranged in such a manner that all rows and all columns have the same sum. Encode such a magic square in an SMT formula (in SMT-LIB, or using the `python` interface).
Which of the following two can be completed to a magical square?

	1	
4		

7		
		8

[4] **4** Solve the following instance of travelling salesman. On the website you will find the file `distances.py`, which lists distances between 13 US cities in miles. Is there a circular route to visit them all below 9000 miles?

The following steps might be helpful:

(a) Create 13 integer variables $c_1, \dots, c_{13}$ with the semantics that the route is $c_1 \rightarrow c_2 \rightarrow \dots c_{13} \rightarrow c_1$, and $c_i = 1$ iff c_i is the first city in the list (New York), $c_i = 2$ iff c_i is the second city in the list (Los Angeles), etc.

(b) Formulate a constraint that the values of all cities are between 1 and 13.

(c) Add a constraint that the values of all cities are different.

(d) Write a function `distance(c_i, c_j)` which takes two city variables and returns and expression for the distance between city c_i and c_j. You can construct this expression as a big if-then-else expression, covering all 13×13 possibilities, looking up distances in the matrix from `distances.py`.

(e) Compute an expression for the total distance of the route by summing up `distance(`c_1`,` c_2`)`, ... `distance(`c_{12}`,` c_{13}`)`, `distance(`c_{13}`,` c_1`)`.

(f) Add a constraint demanding that the total distance is below the given bound.