

SAT and SMT Solving | SS 2018 | LVA 703048

Exercises 2 | March 22, 2018

1 Consider the formula

$$(\neg 1 \vee 2) \wedge (\neg 1 \vee \neg 2 \vee 3) \wedge (\neg 1 \vee \neg 2 \vee \neg 3 \vee 4) \wedge (\neg 2 \vee \neg 5) \wedge (\neg 3 \vee 5 \vee \neg 6) \wedge (\neg 7 \vee 8) \wedge (\neg 8 \vee 9) \wedge (\neg 8 \vee \neg 9 \vee \neg 10) \wedge (\neg 12 \vee 13) \wedge (10 \vee \neg 12 \vee \neg 13 \vee \neg 14) \wedge (\neg 8 \vee \neg 12 \vee \neg 13 \vee 14)$$

and suppose a DPLL inference sequence reached the state $1^d\ 2\ 3\ 4\ \overline{5}\ \overline{6}\ 7^d\ 8\ 9\ \overline{10}\ 11^d\ 12^d\ 13\ \overline{14}$.

[2] (a) Construct a conflict graph. Indicate some cuts which correspond to possible backjump clauses.

[2] (b) Use the approach given on the slides of Week 2 to determine possible backjump clauses by resolution.

[2] 2 Determine satisfiability and validity of the formula

$$(\neg 1 \vee 3) \wedge (5 \vee 2 \vee 3) \wedge (1 \vee \neg 2) \wedge (\neg 3 \vee 4) \wedge (\neg 1 \vee \neg 4) \wedge (1 \vee \neg 3 \vee \neg 4) \wedge (2 \vee \neg 4 \vee 5)$$

using the Python `z3` library. Print a satisfying assignment if the formula is satisfiable.

3 Use the file `hash_stub.py`, which defines a function `mk_string` that returns a bit vector representation of a string (based on the `z3` library).

[1] (a) Define a function `bitwise_xor(a,b)` which takes two bit vectors a and b (as returned by `mk_string`) and returns the bitwise exclusive or of the two bit vectors.

[1] (b) Define functions `left_shift(a,n)` and `right_shift(a,n)` which shift a bit vector a by n bits to the left and to the right, respectively (filling up with 0 bits in both cases).

[2] (c) Use the previously defined functions to find hash collisions for `rotation_hash` as defined in `hash_stub.py` and on the slides of Week 2. To that end, define a function which takes a bit vectors encoding of a string a and returns a 32 bit vector which corresponds to the hash of a. Then, encode equality of two bitvectors of the same length. Use these functions to find two strings of length 5 which have the same rotation hash.

[2] ⋆(d) Encode addition of two bit vectors and use it to find hash collisions of the `shift_add_xor` hashing function defined in `hash_stub.py`.

Exercises marked with a ⋆ are optional. Solving them gives bonus points if you submit them before the course via OLAT or email.