Technische Universität Wien
Fakultät für Informatik
Uwe Egly

WS 2019
FMI UE Exercise Sheet (Block 2)

185.A93: Exercises on Formal Methods in Computer Science

The topic of these exercises related to block 2 (satisfiability) is modeling and solving satisfiability modulo theories (SMT) problems using the theorem prover Z3:

`https://github.com/Z3Prover/z3`

The schedule of the exercises is as follows:

- November 5: Presentation of this exercise sheet and introduction.
- **November 25**: **Submission deadline** (upload in TUWEL).
- December 10: Presentation of solutions.

A brief introduction to SMT and Z3 will be given in the first meeting on November 5 (slides available in TUWEL). The following resources may be helpful:

- Tutorial on Z3 (*highly recommended*):
  `https://rise4fun.com/Z3/tutorial/guide`
- Appendix chapter "SMT-LIB: A Brief Tutorial" (available as PDF from TUWEL) of the book *Daniel Kroening and Ofer Strichman. Decision Procedures - An Algorithmic Point of View, Second Edition.*
- SMT-LIB tutorial by David R. Cok:
  `http://smtlib.github.io/jSMTLIB/SMTLIBTutorial.pdf`

A total number of 15 points can be achieved by solving the exercises on this sheet.
Please consider the following *important guidelines* when preparing and submitting your solutions to the exercises:

- Your submission of the solution to a (sub-)exercise must contain the following two files.
  - A text file containing a list of commands that can be interpreted by Z3 as *input*. Examples of input files and Z3 commands will be provided in the meeting on November 5 when this exercise sheet is presented.

  - A text file containing the *output* of Z3 that is produced for the respective input file.

  Additionally, one pdf file containing your proofs and additional comments may be uploaded.

  Please note that your solution will *not* be considered for grading if either of the first two files is missing.

- Please *explain and comment* each of your solutions *in detail*. Explanations and comments are considered an *essential part* of your submission and are *necessary to receive full points* for an exercise.

  Explanations and comments should be included directly in the submitted input files of Z3 by prefixing the comment lines with a semicolon ";". Any line prefixed with ";" is interpreted as a comment and hence ignored by Z3.

- Proofs and additional comments can be presented in the additional pdf file. Please indicate precisely on which (sub-)exercise you comment on.

- Please submit the solutions to the exercises as *separate text files* by uploading them in TUWEL, and adhere to the following file naming conventions:

  `<surname>-<matrnr>-<exnr>-input.txt`

  `<surname>-<matrnr>-<exnr>-output.txt`

  `<surname>-<matrnr>.pdf`

  where `<surname>` is your surname, `<matrnr>` is your matriculation number, `<exnr>` is the number of the sub-exercise on this sheet, and `input` or `output` indicate whether the file contains the input to or the output from Z3, respectively.

- Please submit your solution *on time* before the deadline. Late submissions will not be considered for grading.

- Please make sure that the input text file `<surname>-<matrnr>-<exnr>-input.txt` is *free of syntactic errors*. Note that your solution will *not* be considered for grading if Z3 reports any syntactic errors.

General information on the organization of this course was presented in the kick-off meeting (slides available in TUWEL).

Exercise 1 Installation of Z3 and Basic Use **1 Point**

Download and install the current version 4.8.6 of Z3 from `https://github.com/Z3Prover/z3`. It is recommended to clone the github repository or to take the zip file with the source code and compile Z3 by yourself (preferably with python support). Read the Z3 tutorial (not all parts are relevant for this exercise sheet). You can find the tutorial at `https://rise4fun.com/Z3/tutorial/guide`.
Given the propositional formulas $\phi_1 := p \to q$ and $\phi_2 := \neg q \to \neg p$, use Z3 to prove that the formula $\phi_1 \leftrightarrow \phi_2$ is valid.

Exercise 2 Propositional Logic: Counterexamples to Validity **1+1 Points**

(a) Use Z3 to show that the propositional formula $\phi := ((x \oplus y) \wedge (y \oplus z)) \to (x \vee z)$ is not valid, where $\oplus$ denotes the XOR operator, and compute a concrete counterexample (i.e., concrete truth assignments to x, y, and z).

(b) Use Z3 and its `get-model` command to iteratively and manually enumerate all counterexamples to the validity of ϕ. When the formula becomes unsatisfiable, retrieve a proof. Explain the generated Z3 proof and provide a resolution proof by yourself (in the pdf file).

Exercise 3 Integer vs. Bit-vector Arithmetic **1+1+1 Points**

(a) Consider line 6 of the `binsearch` program (cf lecture slides): `m = (l + h) / 2`. Recall the use of data type (signed) `int` there. Use Z3 to check whether the assertion $l \leq m \wedge m \leq h$ holds in the theory of bit-vectors of size 32, assuming that $0 \leq l \leq h$. Justify your answer in detail, e.g., by providing a concrete counterexample.

(b) Like (a) but in the theory of integers.

(c) Like (a), but instead of the statement
```
m = (l + h) / 2
```
consider the statement
`m = l + ((h - l) / 2)` and check whether the assertion holds.

Exercise 4 Equality Logic and Uninterpreted Functions **2+2+0.5+0.5 Points**

(a) Consider the *EUF*-formula

$$\varphi^{EUF} := x = y \wedge F(x) = G(y) \wedge z = G(F(y)) \wedge z \neq G(F(x))$$

where F and G denote uninterpreted functions. Use Z3 to check whether φ^{EUF} is *E-satisfiable*. Use the semantic proof method from the lecture and perform a proof of the result by yourself.

(b) Ackermanns's translation as discussed in the lecture is validity-preserving. Develop a variant of the Achermann's translation which is satisfiablility-preserving and prove

its correctness. Apply this variant to φ^{EUF} to obtain the E-formula φ^E and use Z3 to check whether φ^E is E-satisfiable.

(c) Consider the formula φ^E from (b) that results from φ^{EUF} by Ackermann's reduction. Construct a formula φ^E_M from φ^E by removing as many functional consistency constraints as possible such that φ^E_M and φ^E have the same logical status. Use Z3 to check whether φ^E_M is E-satisfiable.

(d) Like (c) but use the unsatisfiable core extraction of Z3 (`get-unsat-core`) to find a *not necessarily minimal subset* of the functional consistency constraints in φ^E.

Exercise 5 Equality Logic and Uninterpreted Predicates **1+1 Points**

Given the EUF-formula

$$\varphi := (y = z \wedge Q(z, x) \wedge P(x, z)) \rightarrow (P(x, y) \wedge Q(x, y))$$

where P and Q denote uninterpreted predicates.

(a) Use Z3 to check whether φ is E-valid.

(b) Like (a) but assume that, in addition to the usual axioms of equality logic, the following axiom is part of the theory: $\forall x, y.\ \big(Q(x, y) \leftrightarrow Q(y, x)\big)$.

Exercise 6 Problem solving with Z3py **1 Point**

Consider numbers with $2n$ decimal digits. How many numbers exist for which the sum of the first n digits equals the sum of the last n digits, if $n = 2, 3, 4$?

Exercise 7 Verifcation of a C-program **1 Point**

```
1 void xInts(unsigned int *x, unsigned int *y)
2 {
3   if(x != y) {
4     *x = *x + *y;
5     *y = *x - *y;
6     *x = *x - *y;
7   }
8 }
```

What is the program fragment supposed to do? Check the correctness of the procedure using Z3. Can we change `unsigned int` to `int`? Check it with Z3 and explain your answer.