Dr. Álvaro Torralba and Prof. Wolfgang Wahlster
Dr. Cosmina Croitoru, Daniel Gnad, Marcel Steinmetz
Yannick Körber, Michael Barz
Christian Bohnenberger, Sophian Guidara, Alexander Rath,
El Khansa Rekik, Julia Wichlacz, Anna Wilhelm

Practical Sheet 3.

Solutions due Tuesday, **June 19**, 23:59, in the Moodle system.

In this sheet, your task is to model different problems in predicate logic, and to solve those models using Z3[1] [2]. To do so, you have to create a text file in Z3's input language, which you can then pass as command line argument to the Z3 executable. The Z3 executable is available in the VM through typing `z3` in a terminal. Printing additional information about the solving process can be done through the `-st` option, e.g., `z3 -st path/to/model.z3`. The result of Z3 (`sat` or `unsat`) will be printed to the console.
Table 1 shows an overview of the Z3 language fragments that are relevant for this sheet. **You must not use any statement that is not included in this table**.

IMPORTANT Please add comments to your models. We will subtract points if you do not put any comments into your models, or the comments provided are insufficient or hard to follow.

Submission instructions Put your `ex3.{z3,log}` and `ex4.{z3,log}` files from Exercises 3 and 4 into an archive called "name1-name2-name3.zip", where "name1", "name2", "name3" are the family names of all authors. Additionally, add a file called "authors.txt" to the archive that contains one line per author, detailing the full name and matriculation number. Please do **not** add any subfolders to your archive. To upload the archive to the Moodle system, go to the corresponding assignment, click "Add submission", and upload it. Note that only one author per group needs to do the submission.

[1] `https://github.com/Z3Prover/z3`
[2] An introduction can be found in `http://rise4fun.com/z3/tutorial/guide`.

Exercise 3. (5 Points)

Prove that if the following three statements hold, then $\forall x \exists y : r(x, y)$, for $x, y, z \in \mathbb{Z}$.

1. $\forall x \exists y : p(x, y)$
2. $\forall x \exists y : q(x, y)$
3. $\forall x \forall y : [(p(x, y) \vee q(x, y)) \implies \forall z : [(p(y, z) \vee q(y, z)) \implies r(x, z)]]$

Model the statements in Z3 in the provided `ex3.z3` file.

Exercise 4. (15 Points)

There are four gardens that cultivate different kinds of flowers (rose, tulip, and lily), vegetables (onion, carrot, and pepper), and fruits (apple, banana, and cherry).
Given that we know the facts given in the list below, use Z3 to prove that there must be (a) a lily in garden1, (b) a fruit in gardens 2 and 3 (the same kind of fruit in both), and (c) tulips and roses are in the same garden. Formulate each of the facts (1.-10.) and the statements to be proved (a-c) as a separate statement in Z3. Write all statements in general form (i.e. without referring to particular objects), unless the description refers to particular objects.

1. Any plant is either a fruit, a flower, or a vegetable.
2. Everybody grows exactly 3 different plants.
3. Every plant is in at least one garden.
4. Exactly one garden has all 3 kinds of fruits.
5. Exactly 3 plants are in 2 or more gardens and they are one vegetable and two fruits. All others plants are in a single garden.
6. There is no garden that grows both rose and carrots.
7. Any garden with tulip has another flower.
8. Garden1 has one fruit, one vegetable and one flower.
9. Garden2 has no flowers.
10. Gardens 1 and 4 have carrots and Garden3 has bananas.

We provide a stub (`ex4.z3`) that you must complete, in which some functions and predicates have already been declared. You are allowed to define more functions or predicates if you need it to complete the solution. The stub also includes some constraints that must be specified, because false statements must be explicitly negated. For example, when specifying which objects are fruits, vegetables, etc, it must also be indicated that all other objects are not fruits (we do this via =).

Note: If you write contradictory axioms, anything becomes provable. To test that this is not the case, try to prove something false (like formula(false)) and see whether the solver also returns Proof found. If that is the case, there must be a contradiction somewhere in your premises.

Statement	Description
General	
`(check-sat)`	Checks whether the predicate logic problem defined up to this point is satisfiable.
`(assert E)`	Adds the boolean / predicate logic expression `E` to the list of formulas to be verified in `check-sat`.
`(get-value (E))`	Prints the value of `E`, where `E` can be an arbitrary expression such as constant, propositions, or boolean combination thereof (must occur after `(check-sat)`).
`(get-model)`	Prints all variable assignments (must occur after `(check-sat)`).
`(echo "message")`	Prints `message` to the console.
`; This is a comment`	Commenting.
Mathematical Expressions	
`(declare-const var Int)`	Defines integer variable `var`.
`var`	Evaluates to the value of variable `var`.
Boolean Expressions	
`true`	Constant for true.
`false`	Constant for false.
`(declare-const p Bool)`	Declares a new proposition with name `p`.
`p`	Evaluates to the truth assignment of `p`
`(not E)`	Evaluates to the negation of `E`.
`(and` E_1 `...` E_n`)`	Conjunction over the expressions E_1 to E_n.
`(or` E_1 `...` E_n`)`	Disjunction over the expressions E_1 to E_n.
`(=` E_1 `...` E_n`)`	Is true if and only if the expressions E_1 to E_n evaluate to the same value. Can be used e.g. to compare variables, or to compute the equivalence over boolean expresions.
`(=>` E E'`)`	Implication, is true if E implies E'.
`(distinct` E_1 `...` E_n`)`	Is true iff every expresion E_1 to E_n evaluates to a different value.
Predicate Logic	
`(declare-fun P (`$T_1 \dots T_n$`) Bool)`	Declares an n-ary predicate with name `P` and parameter types T_1 to T_n.
`(P` t_1 `...` t_n`)`	Evaluates to the value of the n-ary predicate `P` with arguments t_1 to t_n.
`(forall ((`x_1 T_1`) ... (`x_n T_n`)) (E))`	All quantification over n variables: x_1 with type T_1, ..., x_n with type T_n. $x_1, \dots, x_n$ can be used in the expression `E`.
`(exists ((`x_1 T_1`) ... (`x_n T_n`)) (E))`	Existential quantification over n variables: x_1 with type T_1, ..., x_n with type T_n. $x_1, \dots, x_n$ can be used in the expression `E`.

Table 1: Z3 input language fragment you may use for the predicate logic modeling exercises. You may use the data types `Int`, `Bool`, or the ones specified in the template files we provide.