SAARLAND UNIVERSITY ARTIFICIAL INTELLIGENCE
Prof. Jana Koehler
Prof. Jörg Hoffmann

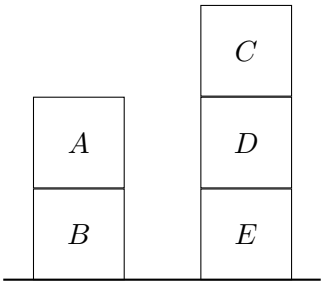
Practical Exercise Sheet 4.

Solutions due Sunday, June 27, 23:59 uploaded in the AI CMS.

This exercise sheet is accompanied with several source files, which we provide through a separate `zip` archive. You can download this archive from the AI CMS under the Materials category.
In this exercise, 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**.

Exercise 1. (4 Points)

In the Blocks World, there are blocks similar to the wooden blocks children play with. The blocks are named and can be stacked to piles[3]. A situation in the Blocks World could look like the following:



We use the following predicates to model the Blocks World:

- $On(x, y)$ holds if and only if x is placed *directly* on top of y. In the example state in the above figure, e.g. $On(A, B)$ holds, but $On(C, E)$ does not.

[1] `https://github.com/Z3Prover/z3`
[2] An introduction can be found in `http://rise4fun.com/z3/tutorial/guide`.
[3] An introduction can be found in `https://en.wikipedia.org/wiki/Blocks_world`

- $Below(x, y)$ holds if and only if x is below y (on the same pile, but not necessarily directly below y). In the example, e.g. $Below(E, C)$ and $Below(B, A)$ holds, but $Below(D, B)$ and $Below(E, A)$ does not.

Solve the following tasks in Z3. Start with the provided template file (`prsh-04-01.z3`). Do not use additional predicates or sorts.

1. Define the following axiom that holds in the Blocks World.
   - $\forall x \forall z \left[Below(x, z) \Leftrightarrow (On(z, x) \vee \exists y \left[On(z, y) \wedge Below(x, y)\right])\right]$ – a block is below a second block if and only if it is directly below that block or if it is below a block that is somewhere below it.
2. Fully specify the state given above by providing the definition of the *On* relation. **Do not define any *Below* relation. We want to derive it via the given axiom.**
3. Show that $Below(E, C)$ holds.

Tip: Introduce several test cases (e.g. *(assert (Below A B)), (assert (Below B A))*) and see what is the result for your model. These will not be part of your assignment, they are just for you to see if everything is fine with the model.

Exercise 2. (6 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 `prsh-04-02.z3` file.

Exercise 3. (10 Points)

Eve, Adam, Paula, and Mike met for dinner. Everybody ordered an appetizer, and a main course. The following meals were ordered: salad (can be ordered for either appetizer or main course), soup (appetizer), cheese plate (appetizer or main course), steak (main course), pasta (main course). In your model, you don't have to distinguish whether a meal is ordered for appetizer, or main course. You can assume that nobody ordered salad or cheese plate for both appetizer and main course. Find out who has ordered what given the following knowledge:

1. Everybody ordered exactly two different meals, one from each category: appetizer, and main course.
2. Every meal is ordered from at least one person.
3. Exactly two meals are ordered from two or more persons. One of those cannot be ordered for appetizer.
4. There is at least one person who ordered two meals that can be ordered for appetizer.
5. Everybody who ordered salad did not order cheese plate.
6. Eve ordered the cheese plate, Adam ordered steak, and Paula ordered soup.

Implement the statements from above into the Z3 template file that we provide (`prsh-04-03.z3`). In this file, we provide you with the types *Person* (eve, adam, ...) and *Meal* (salad, soup, ...), and with the predicates required for modeling the statements from above (if needed, you may however introduce new predicates). Let Z3 solve your model.
Fill the table below with the solution returned from Z3. How can you use Z3 to prove that this is, or is not, the only solution? Give your answers in a separate PDF or text file.

	Appetizer	Main course
Eve		
Adam		
Paula		
Mike		

Submission Instructions

Solutions need to be packaged into a `.zip` file and uploaded in the AI CMS. The `.zip` file has to contain a single folder with name:
`AI2021_PE4_mat1_mat2_mat3` where `mat1, mat2, mat3` are the matriculation numbers of the students who submit together. This folder must contain the following files:

- `authors.txt` listing the names and matriculation numbers of all students who submit together. Use one line per student and no spaces: Name;Matriculation number.
- The Z3 files containing your solutions. **Do not rename** the Z3 template files.

Do not add any other folder or sub folder, this means place all files directly into `AI2021_PE4_mat1_mat2_mat3`. Do not place any file outside of `AI2021_PE4_mat1_mat2_mat3`.

Only one student of each group needs to do the submission! Remember that this sheet can be submitted in groups of up to three members (all members of the group must however be assigned to the same tutorial).

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.