Prof. Dr. Christoph Matheja
Theory of Correct Systems @ UOL
Software Systems Engineering @ DTU

Program Verification – Assignment 2

- Submit your solutions online in groups of 1-3 before the deadline.
- The tasks are numbered consecutively across all assignments. If you have questions, you can thus refer to the task number.
- You can use the concepts and notation from the course material without explanation. Any other definitions or notations must be defined before they are used. Similarly, theorems or mathematical properties that are neither introduced in the course nor well-known from typical undergraduate courses, for example discrete math or logic, need to be introduced and proven first. When in doubt about using a property you know from another course, ask us first.

T5: Encoding stacks (25 Points)

The goal of this exercise is to develop a custom theory for a data type, specifically stacks of integers. The API of our data type consists of the following functions:

- `empty():Stack` constructs the empty stack.
- `push(v:Int, s:Stack):Stack` constructs a new stack by pushing the value `v` on top of the stack `s`.
- `isEmpty(s:Stack):Bool` returns true if `s` is the empty stack, otherwise false.
- `pop(s:Stack):Stack` returns the stack obtained by removing the top element of `s`. It is unspecified what happens if `s` is empty.
- `top(s:Stack):Int` returns the element at the top of the stack `s`. It is unspecified what happens if `s` is empty.
- `sum(s:Stack):Int` returns the sum of all integers in the stack `s`.

(a) *Define a theory for stacks with the above API in Z3. You can use either SMT-LIB or one of the APIs, e.g. Z3Py.*

(b) *Test that your axiom system is indeed satisfiable and that you can prove the following property with Z3.*

```
sum(push(top(pop(push(8, push(7, empty)))),
         pop(push(5, push(9, push(12, empty)))))) == 28
```

(c) *Axiomatize an additional function* `sorted`*, and prove that the previous stack is sorted.*

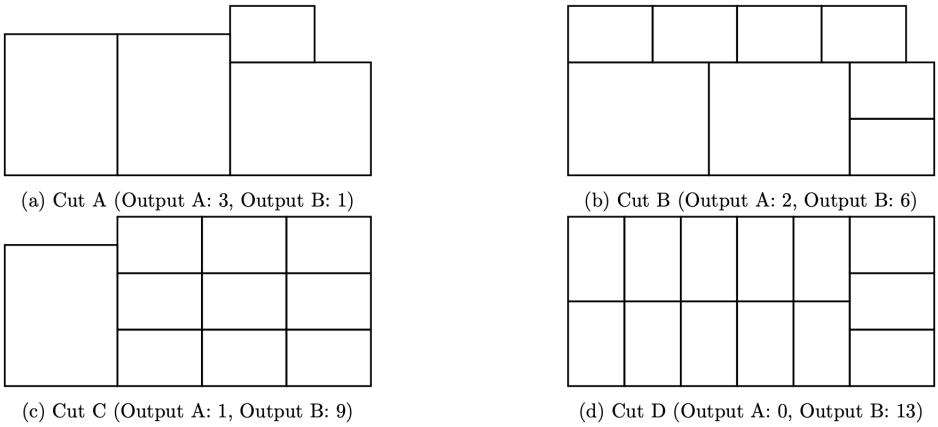
Prof. Dr. Christoph Matheja
Theory of Correct Systems @ UOL
Software Systems Engineering @ DTU

T6: Cutting sheets (25 Points)

SMT solvers are general purpose tools that are useful for other applications than program verification. For example, assume you are in a workshop and have a supply of large rectangular wooden planks (6×13 square meters each). You need to produce smaller planks of size A (4×5 square meters) and size B (2×3 square meters) by cutting your available planks into smaller ones. You can cut a 6×13 sqm plank into planks of size A or B in four ways as illustrated in fig. 1.

(a) *Use Z3 to determine the minimal number of* 6×13 *sqm planks that you need to produce 800 planks of size* A *and 400 planks of size* B*.*

(b) *Additionally, determine how to cut those* 6×13 *sqm planks.*



(a) Cut A (Output A: 3, Output B: 1)

(b) Cut B (Output A: 2, Output B: 6)

(c) Cut C (Output A: 1, Output B: 9)

(d) Cut D (Output A: 0, Output B: 13)

Figure 1: Allowed sheet cuts.

Prof. Dr. Christoph Matheja
Theory of Correct Systems @ UOL
Software Systems Engineering @ DTU

T7: First-order Definability (20 Points)

The lecture material discusses various examples of theories with a decidable SMT problem. However, we sometimes need additional functions that are not supported by those theories. It is thus natural to ask which functions we can add without losing decidability. In this task, we explore this question through the notion of definability.
Let $\mathfrak{A}$ be a Σ-Structure with universe $\mathbf{A}$. Moreover, let $g\colon \mathbf{A}^n \to \mathbf{A}$ be an n-ary function. We call g *definable* in $\mathfrak{A}$ if there exists a $\mathbf{FOL}[\Sigma]$-formula $F(x_1, \ldots, x_n)$ such that for every variable assignment $\mathfrak{v}$,

$$\mathfrak{A}, \mathfrak{v} \models F(x_1, \ldots, x_n, y) \quad \text{iff} \quad g(\mathfrak{v}(x_1), \ldots, \mathfrak{v}(x_n)) = \mathfrak{v}(y).$$

In other words, g can be expressed in first-order logic using the existing vocabulary in Σ. Hence, we can safely use g when working with the theory of $\mathfrak{A}$.

Determine for each of the structures $\mathfrak{A}_i$ and functions g_i below whether g_i is $\mathbf{FOL}[\Sigma]$-definable in $\mathfrak{A}_i$, where each symbol has its canonical meaning (i.e. $\mathbb{N}$ is the set of natural numbers, $\cdot$ is the usual multiplication and so on). To show that the answer is "yes", provide a suitable formula as described above. To show that the answer is "no", prove that such a formula does not exist.

1. $\mathfrak{A}_1 = (\mathbb{N}, +, 0, 1, <)$ and $g_1\colon \mathbb{N} \to \mathbb{N}$, $n \mapsto n \bmod 2$.
2. $\mathfrak{A}_2 = (\mathbb{R}, +, \cdot, 0, 1, <)$ and $g_2\colon \mathbb{R} \to \mathbb{R}$ with $g_2(r) = \lfloor r \rfloor$, i.e. r is rounded *down* to the next integer. For example, $\lfloor 3.14 \rfloor = 3$.

   Note: We updated this task as the original task was harder than intended.
3. $\mathfrak{A}_3 = (\mathbb{N}, 0, 1, +, <)$ and $g_3\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with $g_3(m, n) = k$ if $k = \frac{m}{n}$ and $g_3(m, n) = 0$ otherwise.
4. $\mathfrak{A}_4 = (\mathbb{N}, \cdot)$ and $g_4\colon \mathbb{N} \to \mathbb{N}$ with $g_4(n) = 1$ if n is a prime number and $g_4(n) = 0$ otherwise.

Hint: To show that a suitable formula does not exist, you most likely want to construct a proof by contradiction. To this end you can leverage known (un)decidability results from the lectures: If the theory of a structure $\mathfrak{A}$ is known to be decidable and the theory of $\mathfrak{A}$ extended by some function g is undecidable, then g cannot be definable in $\mathfrak{A}$.

T8: Star Battle (30 Points)

The goal of this exercise is to develop a tool for solving a puzzle game called Star Battle. The game is played on an a square grid that is partitioned into connected regions. Each region is a set of cells that forms one contiguous shape. The objective is to place n stars in the grid according to the following rules:

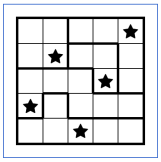
1. Each row contains exactly n stars
2. Each column contains exactly n stars
3. Each region contains exactly n stars
4. No two stars touch each other (not even diagonally)

where n is a parameter of the puzzle. Three examples of grids with their respective solutions are shown in fig. 2.

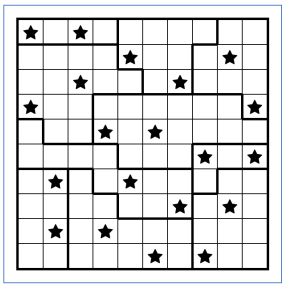
(a) *Implement a solver for n-star battle using any language with a Z3 API that you like (a python template is provided)*

(b) *Check whether your solutions for the grids in fig. 2 match the provided ones.*

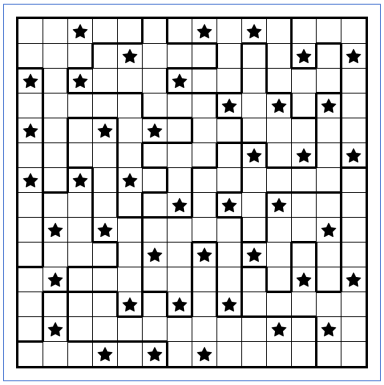
(c) *Use Z3 to check whether there are solutions to the Star Battle puzzle other than the one you found, or to prove that your solution is unique.*



(a) 1-star battle (`GRID_A`)



(b) 2-star battle (`GRID_B`)



(c) 3-star battle (`GRID_C`)

Figure 2: Three filled n-star battle grids.