CS 511
Formal Methods for High-Assurance Software Engineering
Homework Assignment 05

Out: 2 October 2020
Due: Friday, 9 October 2020, by 11:59 pm

- There are six problems in this assignment. The first 4 problems have to be solved by hand. The last 2 problems are implementation/coding problems.
- You should submit a single "`.pdf`" file to Gradescope, where you include links to your scripts for Problem 5 and Problem 6. Your scripts should be stored in the repository of your Github account.
- For full credit in the homework, you need to complete 4 out of 6 problems in this assignment. Each is worth 4 points. Of course, you may want to try all 6 problems. You will get credit for all extra exercises you do (correctly!).

Problem 1 For the model $\mathcal{M} \stackrel{\text{def}}{=} (\mathbb{N}, +, \times)$ where $\mathbb{N}$ is the set of natural numbers:

1. Write a first-order wff $\varphi_1(x)$ with exactly one free variable x which defines the set $\{0\}$.
2. Write a first-order wff $\varphi_2(x)$ with exactly one free variable x which defines the set $\{1\}$.
3. Write a first-order wff $\varphi_3(x, y)$ with exactly two free variables $\{x, y\}$ which defines the set $\{\langle m, n\rangle \mid n = m + 1 \text{ in } \mathbb{N}\}$.
4. Write a first-order wff $\varphi_4(x, y)$ with exactly two free variables $\{x, y\}$ which defines the set $\{\langle m, n\rangle \mid m < n \text{ in } \mathbb{N}\}$.

Problem 2 This problem is extracted from the exercise on page 7 in Lecture Slides 16, ***Prenex Normal Form and Skolemiztion***. Do the three following parts in sequence:

1. Use natural deduction to show that: $(\forall x\, \varphi(x, f(x))) \;\vdash\; (\forall x\, \exists y\, \varphi(x, y))$.
2. Show that: $(\forall x\, \exists y\, \varphi(x, y)) \;\not\models\; (\forall x\, \varphi(x, f(x)))$.

   Hint: It suffices to define one model that makes $(\forall x\, \exists y\, \varphi(x, y))$ *true* and $(\forall x\, \varphi(x, f(x)))$ *false*.
3. Show that: $(\forall x\, \exists y\, \varphi(x, y)) \;\not\vdash\; (\forall x\, \varphi(x, f(x)))$.

   Hint: Use the result of the preceding part, which you assume to have solved correctly.

Problem 3 [LCS, page 163], Exercise 2.4.5.

Problem 4 [LCS, page 163], Exercise 2.4.6.

Problem 5 There are two parts in this problem:

1. Write a script in the conventions of **Prover9** and **Mace4** to find a smallest non-Abelian group G, *i.e.*, a non-Abelian group with the smallest possible number of elements in its domain.
2. After obtaining an output from your script, typeset with Latex the *multiplication table* of G and include it as part of your answer for this problem, *i.e.*, G's *multiplication table* should appear in the pdf file of your submitted assignment.

Problem 6 In this problem, you have to write scripts in the conventions of **Prover9** and **Mace4** to prove that two C-like program fragments, `power3` and `power3_new`, are equivalent. The two fragments are shown in Figure 1. The following first-order wff's, φ_a and φ_b, model the behavior of `power3` and `power3_new`, respectively:

$$\varphi_a \stackrel{\text{def}}{=} (\mathtt{out}_{a,0} \approx \mathtt{in}_{a,0}) \;\wedge\; (\mathtt{out}_{a,1} \approx \mathtt{out}_{a,0} \times \mathtt{in}_{a,0}) \;\wedge\; (\mathtt{out}_{a,2} \approx \mathtt{out}_{a,1} \times \mathtt{in}_{a,0})$$

$$\varphi_b \stackrel{\text{def}}{=} (\mathtt{out}_{b,0} \approx (\mathtt{in}_{b,0} \times \mathtt{in}_{b,0}) \times \mathtt{in}_{b,0})$$

The sets of free variables in each of the two wff's are:

$$\mathrm{FV}(\varphi_a) = \{\mathtt{in}_{a,0},\ \mathtt{out}_{a,0},\ \mathtt{out}_{a,1},\ \mathtt{out}_{a,2}\}$$

$$\mathrm{FV}(\varphi_b) = \{\mathtt{in}_{b,0},\ \mathtt{out}_{b,0}\}$$

The variables $\{\mathtt{in}_a, \mathtt{out}_a, \mathtt{in}_b, \mathtt{out}_b\}$ in the wff's correspond to the memory locations referenced by {`in_a`, `out_a`, `in_b`, `out_b`} in the programs. The second subscripts (natural numbers) attached to the variables in the wff's are *time stamps*; *e.g.*, $\mathtt{out}_{a,0}$, $\mathtt{out}_{a,1}$, and $\mathtt{out}_{a,2}$ correspond to the first value, second value, and third value, stored in location `out_a` during execution of program `power3`.

There are two parts in this problem:

1. Write a script in the conventions of **Prover9** and **Mace4** to prove that `power3` and `power3_new` are equivalent. Specifically, you have to show that:

   $$(\mathtt{in}_{a,0} \approx \mathtt{in}_{b,0}) \wedge \varphi_a \wedge \varphi_b \vdash (\mathtt{out}_{a,2} \approx \mathtt{out}_{b,0})$$

2. Increment the limit of the `for`-loop in `power3` from "`i < 2`" to "`i < 3`". Call the resulting program `power3_inc`. Write a script in the conventions of **Prover9** and **Mace4** to prove that `power3_inc` and `power3_new` are *not* equivalent. Specifically, you have to show that:

   $$(\mathtt{in}_{a,0} \approx \mathtt{in}_{b,0}) \wedge \varphi_a^{\mathtt{inc}} \wedge \varphi_b \nvdash (\mathtt{out}_{a,3} \approx \mathtt{out}_{b,0})$$

   where $\varphi_a^{\mathtt{inc}}$ models the behavior of `power3_inc`. You need to infer wff $\varphi_a^{\mathtt{inc}}$ from the text of `power3_inc` yourself, in the same way that we inferred φ_a from the text of `power3`.

```
int power3 (int in_a)
{
    int i, out_a ;
    out_a = in_a ;
    for (i = 0; i < 2; i++)
      out_a = out_a * in_a ;
    return out_a ;
}
```

```
int power3_new (int in_b)
{
    int out_b ;
    out_b = (in_b * in_b) * in_b ;
    return out_b ;
}
```

Figure 1: Two different implementations of the cubic function, `power3` and `power3_new`.