IA085: Satisfiability and Automated Reasoning

Seminar 4

Exercise 1 *Consider an alphametics summation puzzle such as the following:*

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

where each letter corresponds to a decimal digit, different letters represent different digits, and the leading digits (here S and M) cannot be zero.

Using `pySMT`, write a function `solve_sum` that receives three strings corresponding to the three rows of the puzzle and finds its solution, if one exists.

Exercise 2 *It is possible to represent a function of one variable $f : \mathbb{Z} \to \mathbb{Z}$ symbolically by a term with one free variable x of sort `Int` whose evaluation gives the result of the function.*

Using `pySMT`, write a function `is_injective` that given a term t representing a function f decides whether the function is injective.

Hint: The method *t.substitute*() might come in handy.

With suitable implementation, your the following code should output `True, True, False, False`:

```
from pysmt.shortcuts import Symbol, Ite
from pysmt.typing import INT

x = Symbol("x", INT)

def is_injective(t):
    pass

print(is_injective(x + 1))                # True
print(is_injective(x * 2))                # True
print(is_injective(Ite(x >= 0, x, -x)))  # False
print(is_injective(x * x))                # False
```

Exercise 3 *Consider the following* symbolic transition system $S = (X, I, T)$*:*

$$\begin{aligned} X &= \{x, y\} \\ I &= x = 0 \land y = 10 \\ T &= \Big((x \neq 3 \land (x' = x + 1)) \lor (x = 3 \land (x' = 0))\Big) \land \\ &\quad \Big((x \neq 2 \land (y' = y + 4)) \lor (x = 2 \land (y' = y - 13))\Big) \end{aligned}$$

Implement bounded model checking *algorithm using `pySMT` and use it to show that the transition system does not satisfy the property $P = y \geq 0$.*

In other words, iteratively check satisfiability of the following formula with increasing k:

$$I(X_1) \land \left(\bigwedge_{1 \leq i \leq k-1} T(X_i, X_{i+1}) \right) \land \neg P(X_k)$$

In the notation above, the set of variables X^k is a new copy of the set of variables X that represents the system state in the time k. The formula $I(X_i)$ is the formula I with all variables X replaced by X_k and similarly for $T(X_i, X_{i+1})$ and $P(X_i)$.

The formula states that there is a sequence of k successor states that starts with an initial state and ends with a state that does not satisfy the property.

If any of the formulas for $k \geq 1$ is satisfiable, the system does not satisfy the property.

BONUS: *How can you use incremental* API *of the* SMT *solver to make the algorithm faster?*

Exercise 4 *Consider the following symbolic transition system $S = (X, I, T)$:*

The system differs from the previous one in the last subformula of T.

$$\begin{aligned} X &= \{x, y\} \\ I &= x = 0 \land y = 10 \\ T &= \Big((x \neq 3 \land (x' = x + 1)) \lor (x = 3 \land (x' = 0)) \Big) \land \\ &\quad \Big((x \neq 2 \land (y' = y + 4)) \lor (x = 2 \land (y' = y - 11)) \Big) \end{aligned}$$

*Implement k-*induction *algorithm using* `pySMT` *and use it to show that the transition system satisfies the property $P = y \geq 0$.*

In other words, iteratively check satisfiability of the following two formulas with increasing k:

$$\varphi_{base} = I(X_1) \land \left(\bigwedge_{1 \leq i \leq k-1} T(X_i, X_{i+1}) \right) \land \left(\bigvee_{1 \leq i \leq k} \neg P(X_i) \right) \quad (1)$$

$$\varphi_{ind} = \left(\bigwedge_{1 \leq i \leq k} T(X_i, X_{i+1}) \right) \land \left(\bigwedge_{1 \leq i \leq k} P(X_k) \right) \land \neg P(X_{k+1}) \quad (2)$$

If the first formula is UNSAT, *then the first k states of the system satisfy the property. If the second formula is* UNSAT, *then if any k successor states satisfy the property, also the next state must satisfy the property.*

If both of the formulas are unsatisfiable for some $k \geq 1$, the system does satisfy the property.

BONUS: *Strengthen the assumption of the formula φ_{ind} so that the first k states have to be parwise distinct. Think about how to change the transition system so that the property cannot be proven without the strengthening and can be proven with it.*

BONUS: *How can you use incremental* API *of the* SMT *solver to make the algorithm faster?*