1 Consider the following C program:

```
int main() {
  char i = 0, j = 0, k = 0;

  while (k < 127) {
    i = i + 1;
    j = j + 2;
    k = k + 3;
    // assert(k == (i + j));
  }
  return 0;
}
```

[2] (a) Draw the program graph.

[2] (b) The aim is to apply bounded model checking to determine whether the assertion in comments holds. Define (on paper) the predicates I (initial state), T (transition function), and P (the property $k = i + j$) to check whether the assertion holds.

[2] (c) Write a program that uses these predicates in bounded model checking to determine whether the assertion can be violated after n steps, and test with increasing n. Is there such a bound n?

[2] 2 In a village of monkeys each monkey owns at least two bananas. As the monkeys are well-organised, each tree contains exactly three monkeys. Monkeys are also very friendly, so every monkey has a partner. Encode the problem to SMT-LIB format and find a model. How many monkeys, bananas, and trees are there according to the model? Do you think this is the minimal possible model?

Hint: The file monkeys.smt2 shows how monkeys can own bananas.

[2] 3 Someone who lives in Dreadbury Mansion stole Aunt Agatha's jewelry. Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein. A thief always hates his victim, and is never richer than his victim. Charles hates no one that Aunt Agatha hates. Agatha hates everyone except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone Aunt Agatha hates. No one hates everyone.

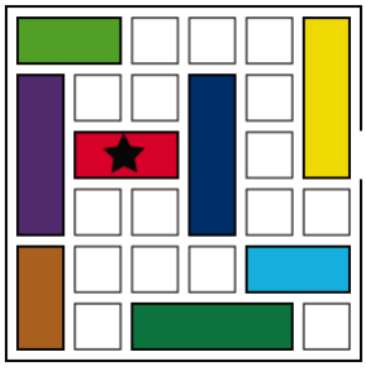
Prove that therefore Agatha stole the jewelry herself.

To do this, show that the following set of formulas is unsatisfiable, by finding a set of Herbrand instances H of the quantified formulas, such that H together with all clauses without variables

is unsatisfiable according to Z3.

$$\mathsf{lives}(\mathsf{agatha}) \wedge \mathsf{lives}(\mathsf{butler}) \wedge \mathsf{lives}(\mathsf{charles})$$
$$\forall x\ y.\ \mathsf{steal\_from}(x, y) \rightarrow \mathsf{hates}(x, y)$$
$$\forall x\ y.\ \mathsf{steal\_from}(x, y) \rightarrow \neg\mathsf{richer}(x, y)$$
$$\forall x.\ \mathsf{hates}(\mathsf{agatha}, x) \rightarrow \neg\mathsf{hates}(\mathsf{charles}, x)$$
$$\mathsf{hates}(\mathsf{agatha}, \mathsf{charles})$$
$$\mathsf{hates}(\mathsf{agatha}, \mathsf{agatha})$$
$$\forall x.\ \mathsf{hates}(\mathsf{agatha}, x) \rightarrow \mathsf{hates}(\mathsf{butler}, x)$$
$$\forall x.\ (\mathsf{lives}(x) \wedge \neg\mathsf{richer}(x, \mathsf{agatha})) \rightarrow \mathsf{hates}(\mathsf{butler}, x)$$
$$\forall x.\ \neg\mathsf{hates}(x, \mathsf{agatha}) \vee \neg\mathsf{hates}(x, \mathsf{butler}) \vee \neg\mathsf{hates}(x, \mathsf{charles})$$
$$\mathsf{steal\_from}(\mathsf{butler}, \mathsf{agatha}) \vee \mathsf{steal\_from}(\mathsf{charles}, \mathsf{agatha}) \qquad \text{(the negated conjecture)}$$

[6] ⋆ 4 In the *Rush Hour* puzzle the goal is to drive the red car out of a jammed parking lot, like in the following picture:



The playing board (i.e., the parking lot) is always a 6×6 grid. A *move* in the game consists of moving one car back or forth onto empty fields (but the cars cannot turn).

Solve the above rush hour puzzle using an SMT encoding and bounded model checking, i.e., by limiting the number of steps.

The following approach might be helpful (though there are many possibilities for an encoding):

- The state, i.e., the current configuration can be described by the x-positions of all horizontal and the y-positions of all vertical cars. So in the example, a state is of the form

$$s = \langle x_{\text{lgreen}},\ x_{\text{red}},\ x_{\text{lblue}},\ x_{\text{dgreen}}, y_{\text{brown}},\ y_{\text{purple}},\ y_{\text{dblue}},\ y_{\text{yellow}} \rangle$$

- It is easy to define a predicate $I(s)$ describing the initial state. In the example, it has to include $y_{\text{brown}} = 1$, $x_{\text{red}} = 2$, etc. (assuming we count from bottom left).
- Define when a cell (X, Y) is *free* in a state s. A cell is free if it is not occupied by any car. For example, (X, Y) is not occupied by the red car if $\neg(x_{\text{red}} \leqslant X < x_{\text{red}} + 2 \wedge Y = 4)$ holds, because $len_{\text{red}} = 2$ and 4 is the (constant) y-position of the red car.
- Using the encoding of a free cell, one can define a predicate $T(s, s')$ to describe a valid transition from state s to state s': In a valid transition every car was moved in a valid way. E.g. for the red car, this can be described as fgollows: if x_{red} in s is different from x'_{red} in s' then the difference cell(s) must have been free in s. (This approach also allows parallel moves, but this is ok.)

- Finally, a success predicate $S(s)$ demanding $x_{\text{red}} = 5$ states that the red car can exit.
- Suppose there are $k+1$ states $s_0, \ldots, s_k$. There is a solution in at most k steps iff

$$I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} S(s_i)$$

  is satisfiable.
- One has to make sure that all states are *valid* in the sense that all its variables are between 1 and $6 - len_{\text{color}}$, where len_{color} is the length of the respective car.

Exercises marked with a $\star$ are optional. Solving them gives bonus points if you submit them before the course via OLAT or email.