universität innsbruck

| Constraint Solving | SS 2024 | LVA 703305 |
|---|---|---|
| Week 13 | | June 14, 2024 |

## Homework

1. Consider the following program for copying the first element of an array to every second position in the array.

```
int a[N];      // some integer array a[0] .. a[N-1] of length N
int i = 0;
while (i < N - 1) {
  a[i + 2] = a[i];
  i = i + 2;  }
```

   (a) Specify the following property as a LIA formula: loop condition implies that the array accesses in the loop body are within bounds. (1 P)

   (b) The formula of part (a) won't be valid, since choosing $N = 6$ and $i = -1$ is a spurious counterexample. Try to find an invariant which rules out those counterexamples where $i$ is too small. Which formulas have to be validated now? (1 P)

   (c) Determine if all of your formulas in part (b) are valid or not. If there is a violating variable assignment, figure out whether it is spurious or not: (1 P)

   - if the assignment can be translated into a program execution that leads to a violation of proper array access (index-out-of-bounds), then correct the program
   - if the assignment is still spurious, so does not correspond to a real program error, indicate how the invariant might be altered to finally prove safety of the array accesses

2. In this exercise the task is to show the expressiveness of array logic. More concrete, you should prove that a combination of linear integer arithmetic and array logic is undecidable.

   To this end, show that termination of a 2-counter-machine program can be encoded as a validity problem of an array logic formula where both $T_I$ and $T_E$ are linear integer arithmetic. (3 P)

   Note: 2-counter-machine programs consist of a finite set of numbered instructions. More formally, there is a finite set of instruction numbers $N \subseteq \mathbb{Z}$, including two distinguished elements `start` and `stop`. For each $n \in N$ that differs from `stop` there is exactly one instruction in the program which either has the form

   $$n\texttt{:}\quad \texttt{c++; goto } m$$

   or

   $$n\texttt{:}\quad \texttt{if c = 0 then c++; goto } m_1\texttt{; else c--; goto } m_2.$$

   Here, $m, m_1, m_2 \in N$ are arbitrary instruction numbers, and `c` is one of two counters `c1` or `c2`. A configuration consists of the current instruction number, and the two values of the two counters `c1` and `c2`, i.e., two integers. The initial configuration is `(start, 0, 0)`, and a configuration is called terminated if the first component is `stop`. Performing a step of a non-terminated configuration is defined as usual. The program is called terminating if from the initial configuration eventually a terminated configuration is reached. The question whether a given 2-counter-machine program terminates is undecidable.

3. (a) A crucial property for verifying a sorting algorithm is that an array $a$ is sorted from start-index $s$ to end-index $e$, i.e., the list $a[s], a[s+1], \ldots, a[e]$ is sorted (w.r.t. $\leq$).

   Define two array logic formulas $\varphi(a, s, e)$ and $\psi(a, s, e)$ that specify that $a$ is sorted from $s$ to $e$. Here, $\varphi$ should only compare array elements of direct neighbors, whereas $\psi$ should encode that an arbitrary array element is smaller than all later array elements. (1 P)

(b) Figure out whether $\varphi(a, s, e)$ and $\psi(a, s, e)$ can also be formulated as array properties as defined on slide 16ff, respectively. So for both formulas provide an equivalent array property or argue why this is not possible. (1 P)

(c) One of the operations in `minsort` is to find a minimum in a given array within a given range specified by start- and end-index, and then move it to the end. A corresponding verification condition for the move looks as follows.

$$\begin{aligned} & \underbrace{(\forall i.s \leq i \leq e \longrightarrow a[m] \leq a[i])}_{m \text{ is position with minimum}} \\ & \wedge x = a[m] \wedge y = a[s] \wedge b = a\{s := x\} \wedge c = b\{m := y\} \\ \longrightarrow & \underbrace{(\forall j.s \leq j \leq e \longrightarrow c[s] \leq c[j])}_{s \text{ is now position with minimum}} \end{aligned}$$

Translate the negated verification condition into an equisatisfiable quantifier-free formula $\varphi$ in LIA + EUF and provide intermediate steps. (2 P)

(d) (Optional bonus exercise) Show unsatisfiability of $\varphi$ of part (c). (2 P)