Formal Methods in Software Engineering WS 2019/20 Thursday 6th February, 2020

Exercise sheet 3

Submission deadline: Sunday 16th February, 2020

Submission: by e-mail to paul.wild@fau.de. Feel free to ask questions!

Merge Sort

In the tutorials, we have implemented a recursive merge sort procedure in F^*, but have not actually shown that it sorts the input list. A code skeleton can be found in the file `MergeSort.fst`. Give the function `merge_sort` the indicated stronger type and add some lemmas with `SMTPat`s to make sure the type check goes through.

Stack

In the tutorials, we have started to implement a stack data structure with references. Finish the implementation in `Stack.fst`. Start at the places marked with `TODO`. Some additional hints:

- The anti-aliasing property of the node list is more naturally stated by making the node list a refinement type. Add the appropriate definition.
- You will probably need to remind the SMT solver that certain references are contained in certain heaplets (`contains h r`).
- You will need some lemmas stating that certain properties of the pre state `h` carry over to the post state `h'` if `modifies_none h h'` holds.
- If your proof fails and you are unsure which parts of the post condition the SMT solver is stuck on, you can add an assertion for some part of it, followed by an `admit()`.