TD 3 - Proving liveness and termination

A Rodin machine is non-blocking if in every execution at least one of its events is always enabled. Equivalently we say that the machine has the liveness property. This can be proved by creating a new invariant which is the disjunction of all guards of the different events of the machine. Proving that the machine is non-blocking is equivalent to proving the invariant.

A Rodin machine terminates if all its executions end up in a final event. To prove termination, one introduces a variant, i.e. an expression with positive integer value and marks all non final events as "convergent". Each execution of a non final event should decrease the value of the variant, and each convergent event generates a corresponding proof obligation.

Exercice 1. Liveness and termination of the counter machine

1. Prove that the AbstractCounter machine is non-blocking.
2. Assuming that reset is a final event, prove that the AbstractCounter machine terminates.
3. Prove that the DecimalCounter machine counter is non-blocking. Does it imply that the AbstractCounter machine is non-blocking? Or is it the other way around?

Exercice 2. Liveness and termination

1. Prove termination and liveness of the MinMax machine.
2. Prove liveness of the ROMController.

Exercice 3. Graphs Load the project available at this url http://www.labri.fr/perso/gimbert/teaching/Rodin/Graph.zip

1. What is the context about?
2. What is the specification about?
3. What does the implementation do?
4. Prove the theorems of the context as well as the correctness of the implementation.

Exercice 4. Searching arrays

1. Create a context ArraySearch and a machine SearchProblem with the following specification : given as input an array $a$ and an integer $x$, decide whether $x$ is stored in $a$.
2. Create a machine LinearSearch that search an element by linear search. Prove correctness, liveness and termination.
3. Create a context SortedArray and a machine DichotomicSearch that
search an element by dichotomy, assuming the array is sorted. Prove
correctness, liveness and termination.

4. Refine the context to specify the initial array size and content and
animate the model with ProB.

Exercice 5. Sorted Arrays  Write some formal specifications and imple-
mentations of the following informal specifications :
1. Checking whether an array is sorted.
2. Checking whether an array is sorted and does not contain twice the
same element.
3. Checking whether an array does contain twice the same element.

Exercice 6. Going further  Check out the examples :
For example

Load the project and read the paper :
http://deploy-eprints.ecs.soton.ac.uk/316/