UNIVERSITY BORDEAUX 1
MASTER INFORMATIC - SOFTWARE ENGINEERING

A PROJECT REPORT
ON

FORMAL DESIGN

BY

NGUYEN Quang Anh

2014-2015

ABSTRACT

Formal Methods in System Design allows the designing, implementing, and validating the correctness of the system. In this project, I implemented and proved the correctness of the machines that check a given array if it's sorted in ascending order, sorted (ascending or descending), check two given arrays if they are identicals, if one array included the other, and sort a given array in ascending order

The purpose of this paper, is to deliver the process of proving the proof obligations, those that was proved interactively, as well as the reasoning for the unproved ones.

1 INTRODUCTION

In the last lesson of Formal Design, we were given a project to finish. Our job is to design several machines that :

- check if a given array is sorted in ascending order
- check if a given array is sorted (ascending or descending order)
- check if two given arrays are identical
- given two arrays, check if one array included the other one
- given an array with pairwise distinct values, sort this array in ascending order
- given an array, sort this array in ascending order

I used a software, named **Rodin**, to do this project. Some theories will be proved in this paper, if cannot be proved in the program.

2 Check if array is sorted in ascending order

- Specification: TestAscendingMachine
- Implementation: TestAscendingMachineImplementation

The only PO that need to be proved interactively, is **liveness/WD** int **TestAscendingMachineImplementation**. I proved this using tthe tactic **Disjunction to implication**

3 Check if array is sorted

- Specification: TestSortedMachine
- Implementation: TestSortedMachineImplementation

The POs that need to be proved interactively is :

- inv9/WD
- liveness/WD
- liveness/THM
- INITIALISATION/inv7/INV
- IS_SORTED/grd1/GRD
- LOOP_EQUAL/inv9/INV

The only tatic that I used, is **Disjuction to implication**

4 Check if two given arrays are identical

- Specification: CompareArraysMachine
- Implementation: CompareArraysMachineImplementation

The POs that need to be proved interactively:

- In CompareArraysMachine
  - NOT_IDENTICAL/grd1/WD
- In CompareArraysMachineImplementation
  - liveness/WD
  - IS_IDENTICAL/grd1/GRD
  - NOT_IDENTICAL/grd1/WD

The only tactic used to solve, is **Disjuction to implication**

5 Check if the values of one array is included in another one

- Specification: ValueIncludedMachine
- Implementation:
  - ValueIncludedMachineImplementation
  - ValueIncludedMachineImplementation2

ValueIncludedMachineImplementation is a machine that check the values of the two arrays whether they satisfy :

$$\forall i \cdot i \in dom(array1) \Rightarrow (\exists j \cdot j \in dom(array2) \Rightarrow array1(i) = array2(j))$$

The complexity of this problem is:

$$O(n_1, n_2) = n_1.n_2 \quad n_1 = card(dom(array1)) \wedge n_2 = card(dom(array2))$$

ValueIncludedMachineImplementation2 is a special case of **ValueIncludedMachineImplementation**, where the two given arrays are sorted in ascending order.

The POs that need to be proved interactively are:

- liveness/WD (in both of the implementations)
- liveness/THM (in both of the implementations)

The tactic that was used to prove, is **Disjunction to implication**