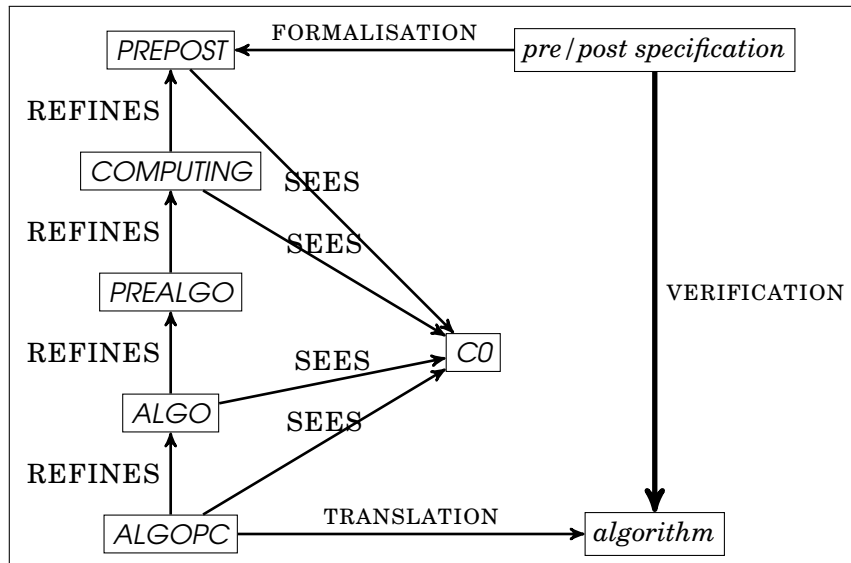


Exercise 1 *fx1-tut2.zip*

We consider a finite sequence of integers v_1, \dots, v_n where n is the length of the sequence and is supposed to be fixed. Write an Event B specification modelling the computation of the value of the summation of the sequence v . You should define carefully v , n and the summation of a finite sequence of integers.

Exercise 2 *fx2-tut2.zip*



Apply the pattern for computing the value n^2 using the sequence $(n+1)^2 = n^2 + n + n + 1$. Write a C function with annotation that you will check with Frama-c.

Exercise 3 *fx3-tut2.zip*

Develop an algorithmic solution with the pattern for the problem of finding the number of occurrences of a value v value v satisfying a condition CO in a table t of dimension n . dimension n . The table is assumed to have a value in an envelope V . seems V and that CO is a part of V .

Exercise 4 *fx4-tut2.zip*

Apply this pattern to find the index i of t such that $t(i) = v$. Write a C function that you will check with Frama-c.

Exercise 5 *fx5-tut2.zip*

Apply this pattern to compute x^3 using $(i+1)^3 = i^3 + 3i^2 + 3i + 1$. We use the following sequences :

- $z_0 = 0$ et $\forall n \in \mathbb{N} : z_{n+1} = z_n + v_n + w_n$
- $v_0 = 0$ et $\forall n \in \mathbb{N} : v_{n+1} = v_n + t_n$
- $t_0 = 3$ et $\forall n \in \mathbb{N} : t_{n+1} = t_n + 6$
- $w_0 = 1$ et $\forall n \in \mathbb{N} : w_{n+1} = w_n + 3$
- $u_0 = 0$ et $\forall n \in \mathbb{N} : u_{n+1} = u_n + 1$

$$\begin{pmatrix} z(i+1) \\ v(i+1) \\ t(i+1) \\ w(i+1) \\ u(i+1) \end{pmatrix} = \begin{pmatrix} z_i + v_i + w_i \\ v_i + t_i \\ t_i + 6 \\ w_i + 3 \\ u_i + 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z(i) \\ v(i) \\ t(i) \\ w(i) \\ u(i) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 6 \\ 3 \\ 1 \end{pmatrix}$$

Write a C function from the development and use Frama-c for checking it.

Exercise 6 (tutorial in Maynooth NUI)

The objective is to design a correct by construction algorithm in a programming language with annotations and proof tool as Frama-c or Dafny.

The problem to solve is the power function defined usually as follows in a classical mathematical language $\text{power} = \lambda x, y. x^y$. For instance, the notation x^y is found in the C programming language and a function can be called for computing the power function.

Question 6.1 Define an inductive statement of the power function by defining the sequence $p(x \mapsto y)$ where $x, y \in \mathbb{N}$. The function $\text{powereb} = \lambda x, y. x^y$ is defined in the Event-B language and you should prove that $\forall x, y \in \mathbb{N}. p(x \mapsto y) = x^y$.

Question 6.2 Define a context POW0 and a prepost machine POW1 expressing the contract of the problem to solve. The problem is to define first the contract in your programming language.

Question 6.3 Develop a computing process according to the methodology of the refinement to get an algorithm.

Question 6.4 Check that the algorithm `power` satisfies the contract using the proof tool related to your programming language. You can use the Event-B machines for deriving a possible loop invariant.

Question 6.5 Using the mathematical expression of the power function namely mathpower that you have defined in POW0, you can derive a recursive algorithm `int p(int a0, int b0)` which is using the inductive definition for computing the function but following a recursive schema. Derive an algorithm in your pet programming language and check the contract.

Question 6.6 The two resulting algorithm `power` and `p` are computing the same value for the same inputs. Using your pet programming language, write a function `check` which is calling `power` and `p` and returning 1 if the two values are equal. Write a contract for expressing this equivalence.

Question 6.7 a and b are two natural numbers and if we launch the function `power` for some values we obtain the following results :

- $\text{power}(0, 0) = 1$
- $\text{power}(0, 1) = 1$
- $\text{power}(0, 2) = 1$
-
- $\text{power}(2, 0) = 1$
-
- $\text{power}(2, 1) = 2$
-
- $\text{power}(2, 2) = 4$

Hence, we obtain the following property $\forall y. y \in \mathbb{N} \Rightarrow \text{power}(0, y) = 1$. However, what does mean $\text{power}(a, b)$?

$\text{power}(a, b) = \underbrace{a \times \dots \times a}_{b \text{ times}}$ and

- $\text{power}(a, 1) = a$
- $\text{power}(a, 2) = a \times a$
- $\text{power}(a, b+c) = \text{power}(a, b) \times \text{power}(a, c)$
- $\text{power}(a, 1) = \text{power}(a, 0+1) = \text{power}(a, 0) \times \text{power}(a, 1) = \text{power}(a, 0) \times a = a : \text{power}(a, 0) = 1, \text{ when } a \neq 0.$

Hence $\forall y. y \in \mathbb{N} \wedge y \neq 0 \Rightarrow \text{power}(y, 0) = 1$.

When $a = 0$, we can give the following properties :

- $\text{power}(0, 1) = 0$

- $power(0, 2) = 0 \times 0 = 0$
- $power(0, 0)$ means that there is no number or is developed as follow : $power(0, 1) = power(0, 0+1) : power(0, 0) \times power(0, 1)$ and $power(0, 0) = 1$.

We summarize as follow :

- $\forall y. y \in \mathbb{N} \wedge y \neq 0 \Rightarrow power(y, 0) = 1$.
- $\forall y. y \in \mathbb{N} \wedge y \neq 0 \Rightarrow power(0, y) = 1$.
- $power$ is not defined for $(0, 0)$.

Write two new theorems in POW0 for stating the two first theorems.

Write the same theorems for the function $\lambda x, y. x^y$ and write the following properties

$\forall y. y \in \mathbb{N} \Rightarrow y^0 = 1$ $\forall y. y \in \mathbb{N} \Rightarrow 0^y = 0$ $0^0 = 1$ $0^0 = 0$ $1 = 2$

What is your conclusion ?

Modify your context POW0 according to the sound definition and modify the contract.