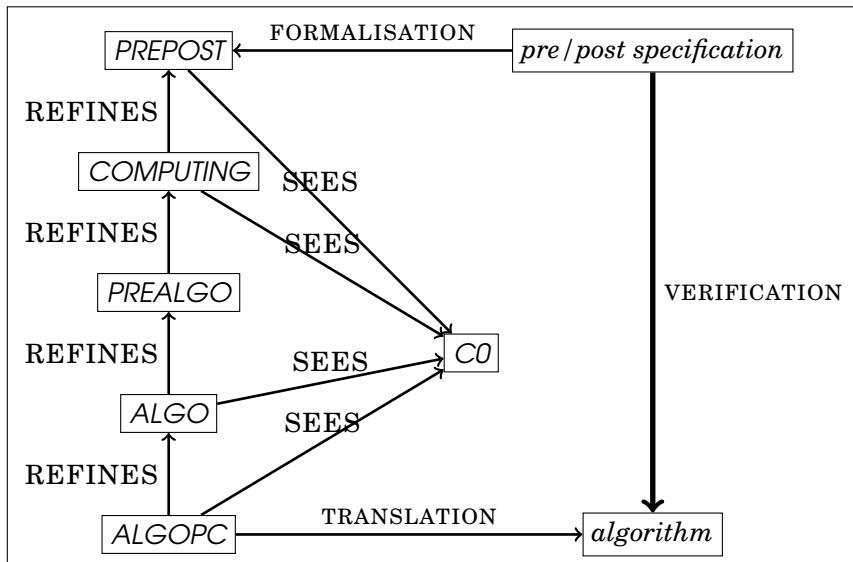


Exercice 1 (*mcfsi3-1.zip*)

Soit une suite de valeurs entières v_1, \dots, v_n où le nombre n est fixé. On souhaite développer un algorithme qui réalise la somme des éléments du vecteur v . Pour cela, on utilisera le patron ci-dessous dont l'archive est sur le site arche sous le nom *iterativepattern.zip* et on commencera par déterminer le contexte $C0$ de ce problème puis on modifiera les différents composants pour construire un développement. Dans ce cas, la solution présentée dans *ex21-fullsummation.zip* définit une suite u qui sera ensuite calculée dans la suite uu .



Question 1.1 Ecrire la pré et post specification de ce problème et définir les notations auxiliaires nécessaires pour une expression en Event-B.

Question 1.2 Définir les deux composants $C0$ et $PREPOST$

Question 1.3 Poursuivre le développement selon le patron notamment en définissant une machine raffinant la machine $PREPOST$ et en introduisant une variable de modèle conservant l'histoire du calcul de la suite principale ?

Question 1.4 Utiliser les règles de transformations des événements pour proposer un algorithme et utiliser Frama-c pour vérifier le programme obtenu.

Exercice 2 (*mcfsi3-ex2.zip* ou *mcfsi3-ex2-plugin.zip*)

Question 2.1 Appliquer ce patron pour calculer la valeur de n^2 en définissant une suite v à l'aide de l'identité suivante $(n+1)^2 = n^2 + n + n + 1$. On pourra utiliser deux variantes avec soit l'introduction de la variable *ok* soit uniquement la variable *pc* en fin de raffinement.

Question 2.2 *power2.c* et *power2.h*

Ecrire une fonction C que vous vérifierez avec Frama-c en prenant soin de définir le contrat et l'invariant de boucle à l'aide des machines précédemment construites.

Exercice 3 *ex-occur.zip* ou *mcfsi3-ex3.zip*

Cet exercice vise à compter le nombre d'occurrences d'une valeur donnée satisfaisant une propriété dans un ensemble de valeurs.

- S est un ensemble d'animaux d'un zoo, A est un ensemble d'animaux vus par un visiteur et P est la propriété les animaux de P sont des singes.
- S est l'ensemble des tableaux de valeurs entières, A est l'ensemble des valeurs continues dans un tableau t de dimension n et P est la propriété les valeurs entières paires contenues dans un tableau donné

Dans cette question, on considère un tableau t de valeurs entières de dimension n et une propriété définie par CO telle que $x \in CO$ signifie que x a la propriété CO .

Développer une solution algorithmique avec le patron pour le problème de la recherche du nombre d'occurrences d'une valeur v satisfaisant une condition CO dans une table t de dimension n . On suppose que le tableau est à valeur dans un ensemble V et que CO est une partie de V .

Exercice 4 ex-search.zip ou mcfsi3-ex4

Une forme plus générale est de considérer un ensemble de valeurs possibles S , un ensemble de données D et une propriété P . Une variante plus générale est la recherche du nombre d'occurrences communes à un ensemble D et un ensemble P qui sont des parties de S et on peut donc caractériser la solution par l'expression cardinalité($D \cap P$).

Appliquer ce patron pour rechercher cardinalité($D \cap P$). Ecrire une fonction C que vous vérifiez avec Frama-c. Pour cela on se ramènera au problème plus général précédent.

Exercice 5 (ex-primrec.zip ou mcfsi3-ex5)

Une fonction primitive récursive f sur les naturels Nat est définie comme suit :

$$\begin{cases} f(x, 0) = g(x) \\ f(x, suc(y)) = h(x, y, f(x, y)) \end{cases}$$

On suppose que g et h sont deux fonctions primitives récursives aussi mais connues.

Question 5.1 Ecrire un premier modèle spécifiant le calcul de f pour une donnée.

Question 5.2 Proposer un raffinement de ce modèle en utilisant les équations et les fonctions g et h .

Question 5.3 Dériver un algorithme par transformation du modèle.

Exercice 6 mcfsi2-ex6

Appliquer le patron pour le cas du calcul de x^3 en utilisant $(i+1)^3 = i^3 + 3i^2 + 3i + 1$. Nous utilisons en fait ces suites :

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

$$\begin{pmatrix} z(i+1) \\ v(i+1) \\ t(i+1) \\ w(i+1) \\ u(i+1) \end{pmatrix} = \begin{pmatrix} z_i + v_i + w_i \\ v(i) + t(i) \\ t(i) + 6 \\ w(i) + 3 \\ u(i) + 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 01 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z(i) \\ v(i) \\ t(i) \\ w(i) \\ u(i) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 6 \\ 3 \\ 1 \end{pmatrix}$$

Ecrire une fonction C que vous vérifierez avec Frama-c.