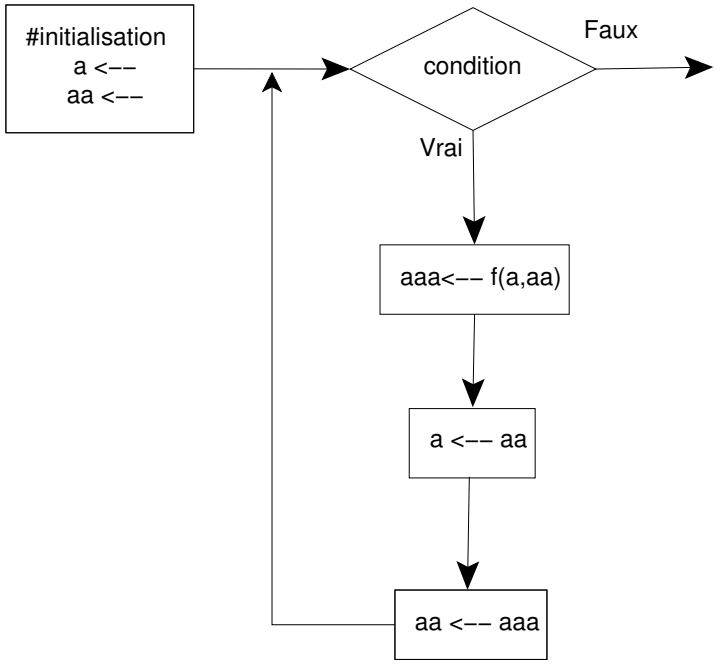

FIG. 1 – Suite récurrente

L'objet de ce texte est de dégager une implémentation minimale d'une suite récurrente ainsi que de présenter la syntaxe Maple d'une boucle "for".

Suite récurrente

Comment implémenter une suite définie par une recurrence d'ordre $p \geq 2$? Une telle suite vérifie

$$\begin{cases} a_0, a_1, \cdots, a_{p-1} \text{ donnés} \\ \forall n \in \mathbb{N} : a_{n+p} = f(a_n, a_{n+1}, \cdots, a_{n+p-1}) \end{cases}$$

où p est un entier fixé et f une fonction de $\mathbb{R}^p$ dans $\mathbb{R}$ fixée ?
Pour implémenter le calcul d'une telle suite, $p+1$ variables sont nécessaires.
Par exemple pour $p = 2$, on peut utiliser des variables `a`, `aa`, `aaa` selon le schéma de la figure 1. Il faut évaluer `condition` à `Vrai` ou `Faux` pour calculer ou non le terme

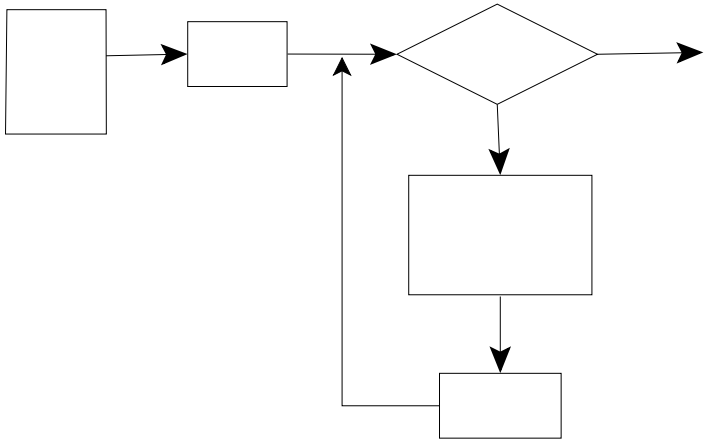

FIG. 2 – Calcul du terme d'indice k.

suivant de la suite. Cette condition (ou test d'arrêt) peut porter sur un indice ou sur une valeur. On peut aussi vouloir garder trace de tous les termes de la suite.

- Pour calculer le terme d'indice k, introduire un compteur et l'incrémenter comme dans la figure 2. Après la sortie de la boucle, quelles valeurs désignent `a`, `aa`, `aaa` et `i` ?
- Un test d'arrêt peut porter sur la différence entre deux valeurs consécutives. On peut prendre par exemple `abs(a-aa) <= eps` où `eps` désigne un nombre décimal (type `float`) petit. Attention, comme par défaut Maple fait des calculs formels exacts il peut être nécessaire d'utiliser un `evalf`.
- Si on veut garder toutes les valeurs intermédiaires, on peut utiliser une séquence (figure 3) ou un tableau.

Boucle "for"

Une boucle "for" est une boucle "while" particulière. Elle contient un compteur entier et on sait à l'avance pour quels entiers consécutifs on doit exécuter la boucle. La figure 2 en est un exemple. On peut l'implémenter en syntaxe Maple par :

```
a:= a0;
aa:= a1;
for i from 1 to k do
  aaa := f(a,aa);
```

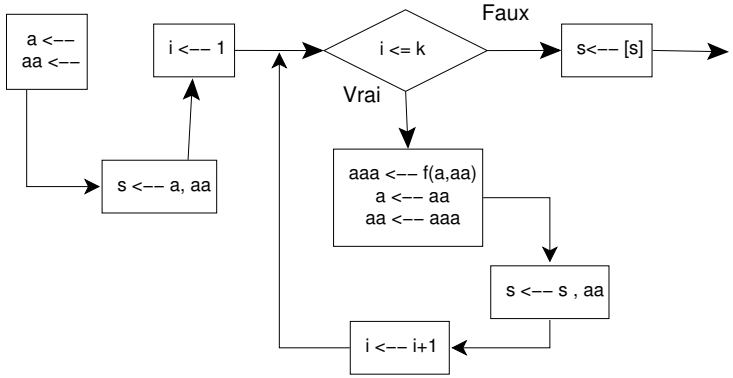

FIG. 3 – Insertion des valeurs dans une séquence

```
  a:= aa;
  aa:= aaa;
od;
```