Rapport du projet Suites Logiques

Mamadou Dian DIALLO & Thierno Ibrahima Sory DIALLO

07 Avril 2018

Ce mini-projet consiste à écrire un ensemble de prédicats qui va nous permettre de compléter quelques suites logiques. Trois suites sont proposées(S1, S2 et S3), donc il faut écrire les prédicats pour compléter ces suites sachant qu'on donne une liste de propositions qui n'est pas toujours valide.

Pour compléter chaque suite, nous avons penser à un prédicat de la forme suiteS1(S1,P,R) pour la suite S1, suiteS2(S2,P,R) pour S2 et suiteS3(S3,P,R) pour S3. Le prédicat suiteS1(S1,P,R) est vrai si et seulement si S1 est une suite, P une liste de propositions et R la suite initiale à laquelle on a rajouté les propositions justes, pareil pour suiteS2(S2,P,R) et suiteS3(S3,P,R). Mais l'objectif du projet est d'avoir un prédicat général suite(S,P,R) qui va appeler chaque prédicat suiteS1(S,P,R), suiteS2(S,P,R), suiteS3(S,P,R) un seul appel n'echouera pas en fonction de la nature de la suite S(S1, S2 ou S3).

Dans l'énoncé, il nous est dit que pour comprendre les suites, il faut observer leur dérivée première.

S1 : 1 2 4 5 7 . . . , dérivée première de S1 : 1 2 1 2

S2 : 1 2 4 7 11 . . . , dérivée première de S2 : 1 2 3 4

S3 : 1 2 7 32 157 . . . , dérivée première de S3 : 1 5 25 125

Toute suite Sn est composée d'un premier élément U1 et d'un élément Un.

On remarque que toutes les suites ont le même premier élément U1 = 1.

Si on essaye de trouver quelques propositions à la main qui respectent les dérivées premières on aura :

Pour S1 : 1 2 4 5 7 8 10 11 13 14 16 17 19 20 22

U1 = 1, U2 = 2, U3 = 4, U4 = 5, U5 = 7, U6 = 8,. . .

On remarque que U2 = U1+U1=2U1, U3=U2+2U1,U4=U3+U1,U5=U4+2U1,U6=U5+U1

On remarque aussi qu'à chaque fois, pour avoir l'élément suivant, on prend l'élément courant et on lui rajoute soit U1, soit 2U1, en fonction de la parité de sa position dans la suite. Si elle est de rang pair, on lui rajoute 2U1, sinon on lui rajoute U1.

Donc la formule pour S1 sera : U1 = 1 et pour tout entier n supérieur à 0, si n est pair alors Un+1 = Un+2U1, sinon Un+1 = Un+U1.

Pour S2 : 1 2 4 7 11 16 22 29 37 46 56
U1 = 1, U2 = 2, U3 = 4, U4 = 7, U5 = 11, U6 = 16, U7 = 22, U8=29, U9=37, U10=46, U11=56

On remarque aussi que U2=U1+1,U3=U2+2,U4=U3+3,U5=U4+4,U6=U5+5,U7=U6+6,U8=U7+7,. . .

Donc l'élément suivant Un+1 est obtenu avec l'élément courant Un + sa position n dans la suite.

Pour S3 : 1 2 7 32 157 782 3907

U1 = 1, U2 = 2, U3 = 7, U4 = 32, U5 = 157, U6 = 782, U7 = 3907

On remarque aussi que Un+1 = Un + 5 à la puissance n - 1

Pour compléter ces suites, on aura besoin des prédicats qui renvoient : le dernier élément d'une liste, l'élément qui se trouve à une position donnée, la position d'un élément dans une liste, la suite obtenue en rajoutant X à la fin d'une liste L, le résultat d'un entier X élévé à une puissance Y.

Réalisation des différents prédicats intermédiaires.

dernierElement(L,X) est vrai si et seulement si X est le dernier élément de la liste L.

dernierElement([X],X).

dernierElement([_|L],R):-dernierElement(L,R).

dernierElement([1,2,3,4,5,6],X).

X = 6

dernierElement([1,2,4,9,12],12).

true

trouveElementPos(L,P,E) est vrai si et seulement si E est l'élément qui se trouve à l'indice P dans la liste L. On suppose que E n'a pas de doublons dans la liste L et que les indices commencent à partir de 1.

trouveElementPos([E|_],1,E).

trouveElementPos([_|L],P,R):- P1 is P-1,trouveElementPos(L,P1,R).

trouveElementPos([1,2,3,4,5,6],6,X).

X = 6

trouveElementPos([1,2,3,4,5,6],3,3).

true

position(E,L,P) est vrai si et seulement si P est la position de E dans la liste L.

On suppose que E n'a pas de doublons dans la liste L et que les indices commencent à partir de 1.

position(E,[E|_],1).

position(E,[_|L], N):- position(E,L,N1),N is N1+1.

position(4,[1,2,3,4],X).

X = 4

position(8,[1,4,8,9],3).

true

ajoutFin(L,X,R) est vrai si et seulement si la liste R est obtenue en ajoutant X à la fin de la liste L.

ajoutFin([ ],X,[X]).

ajoutFin([E|L],X,[E|R]):-ajoutFin(L,X,R).

ajoutFin([1,2,3,4,5,6],7,R).

R = [1,2,3,4,5,6,7]

puissance(X,Y,R) est vrai si et seulement si R est égal à X élévé à la puissance Y.

puissance(_,0,1).

puissance(X,Y, R):- Y1 is Y-1, puissance(X,Y1,R1),R is X*R1.

puissance(3,1,R).

R = 3

puissance(2,3,8).

true

Définition du prédicat suiteS1

suiteS1(S,P,R) est vrai si et seulement si S est une suite qui a pour dérivé première 1 2 1 2 ..... 1 2 1 2, P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes.

suiteS1(L,[ ],L).

suiteS1(L,[E|F], R):- dernierElement(L,X), position(X,L,P), 1 is P mod 2,

```
trouveElementPos(L,1,T), E is X+T , ajoutFin(L,E,W),

suiteS1(W,F, R).
```

suiteS1(L,[E|F], R):- dernierElement(L,X), position(X,L,P), 1 is P mod 2,

```
trouveElementPos(L,1,T), E \== X+T, suiteS1(L,F, R).
```

suiteS1(L,[E|F], R):- dernierElement(L,X), position(X,L,P), 0 is P mod 2,

```
trouveElementPos(L,1,T), Z is 2*T, E is X+Z,

ajoutFin(L,E,W),suiteS1(W,F, R).
```

suiteS1(L,[E|F], R):- dernierElement(L,X), position(X,L,P), 0 is P mod 2,

```
trouveElementPos(L,1,T), Z is 2*T, E \== X+Z,

suiteS1(L,F, R).
```

suiteS1([1,2,4,5],[7,8,9,10,11,12],R).

R = [1,2,4,5,7,8,10,11]

suiteS1([1,2,4,5],[6,7,8,9,10,11,12,13],[1,2,4,5,7,8,10,11,13]).

true

Définition du prédicat suiteS2

suiteS2(S,P,R) est vrai si et seulement si S est une suite qui a pour dérivé première 1 2 3 ... n-1, n étant le nombre d'éléments de S, P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes.

suiteS2(L,[ ],L).

suiteS2(L,[E|F], R):- dernierElement(L,X), position(X,L,P), E is X+P,

```
ajoutFin(L,E,W),suiteS2(W,F, R).
```

suiteS2(L,[E|F], R):- dernierElement(L,X), position(X,L,P),

```
E \== X+P, suiteS2(L,F, R).
```

suiteS2([1,2,4,7],[11,16,22,23],S).

S = [1,2,4,7,11,16,22]

Définition du prédicat suiteS3

suiteS3(S,P,R) est vrai si et seulement si S est une suite dont la dérivé multiplicative de la dérivé première est une suite constante(5 5 5 ... 5 5 5) , P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes.

suiteS3(L,[ ],L).

suiteS3(L,[E|F], R):- dernierElement(L,X), position(X,L,P), P1 is P-1,

```
puissance(5,P1,R), E is X+R ,

ajoutFin(L,E,W),suiteS3(W,F, R).
```

---

suiteS3(L,[E|F], R):- dernierElement(L,X), position(X,L,P), P1 is P-1,

```
puissance(5,P1,R), E \== X+R,

suiteS3(L,F, R).
```

suiteS3([1,2,7], [32,40,157,289,782], L).

Quand on essaye d'exécuter ce prédicat, il nous ramène Out of local stack et on n'a pas pu comprendre pourquoi le programme ne terminait pas.

Le résultat devrait être L = [1,2,7,32,157,782]

Nous avons rajouté trois autres prédicats pour enrichir le projet

Définition du prédicat suiteS4

suiteS4(S,P,R) est vrai si et seulement si S est une suite dont les elements sont obtenus avec Un+1 = Un * (Un + 1) , P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes.

suiteS4(L,[ ],L).

suiteS4(L,[E|F],R) :- dernierElement(L,X), E is X*(X+1),

```
ajoutFin(L,E,W),suiteS4(W,F,R).
```

suiteS4(L,[E|F],R) :- dernierElement(L,X), E == X*(X+1),

```
suiteS4(L,F,R).
```

suiteS4([1,2,6], [42,52,100,1806], L).

L = [1,2,6,42,1806]

suiteS4([1,2,6], [42,52,100,1806], [1,2,6,1806]).

false

Définition du prédicat suiteS5

suiteS5(S,P,R) est vrai si et seulement si S est une suite dont les elements sont obtenus avec Un+1 = Un + 4, P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes.

suiteS5(L,[ ],L).

suiteS5(L,[E|F],R) :- dernierElement(L,X), E is X + 4,

```
ajoutFin(L,E,W), suiteS5(W,F,R).
```

suiteS5(L,[E|F],R) :- dernierElement(L,X), E == X + 4,

```
suiteS5(L,F,R).
```

suiteS5([1,5,9,13], [17,21,22,23,25,26,28,29], L).

L = [1,5,9,13,17,21,25,29]

suiteS5([2,6,10,14], [18,22,24,26,27,29,30,34,38,40], [2,6,10,14,18,22,26,30,34,38]).

true

suiteS5([1,5,9,13], [17,21,22,23,25,26,28,29], [1,5,9,13,21,25,29]).

false

Définition du prédicat suiteS6

suiteS6(S,P,R) est vrai si et seulement si S est une suite et à partir de la troisième position n = 3, Un+1 = Un + Un-1, P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes.

suiteS6(L,[ ],L).

suiteS6(L,[E|F],R) :- dernierElement(L,X), position(X,L,P), trouveElementPos(L,P-1,Y), E is X+Y, ajoutFin(L,E,W), suiteS6(W,F,R).

suiteS6(L,[E|F],R) :- dernierElement(L,X), position(X,L,P), trouveElementPos(L,P-1,Y),

```
E \== X+Y, suiteS6(L,F,R).
```

suiteS6([1,2,3,5], [8,13,14,15,21,22,34,35], L).

L = [1, 2, 3, 5, 8, 13, 21, 34]

suiteS6([2,4,6,10,16], [26,42,68,70,110,150,178], [2,4,6,10,16,26,42,68,110,178]).

true

suiteS6([2,4,6,10,16], [26,32], [2,4,6,10,16,26,42]).

false

Définition du prédicat général suite(S,P,R) qui appelle tous les autres en fonction des différentes suites

suite(S,P,R) est vrai si et seulement si S est une suite, P une liste croissante de propositions et R est obtenue en concatenant la liste S avec les propositions qui sont justes. Dans ce prédicat, on utilise tous les prédicats définis pour les différentes suites S1, S2,.. Si avec suiteS1 le prédicat échoue, on réessaye l'éxécution avec suiteS2, ainsi de suite.

suite(S,P,R) :- suiteS1(S,P,R).

suite(S,P,R) :- suiteS2(S,P,R).

suite(S,P,R) :- suiteS4(S,P,R).

suite(S,P,R) :- suiteS5(S,P,R).

suite(S,P,R) :- suiteS6(S,P,R).

Test de la suite S1 : suite([1,2,4,5], [7,8,9,10,11,12], R).

R = [1,2,4,5,7,8,10,11]

Test de la suite S2 : suite([1,2,4,7], [11,16,22,23], S).

S = [1,2,4,7,11,16,22]

Test de la suite S4 : suite([1,2,6], [42,52,100,1806], L).

L = [1, 2, 6, 42, 1806]

Test de la suite S5 : suite([2,6,10,14], [18,22,24,26,27,29,30,34,38,40], L).

L = [2, 6, 10, 14, 18, 22, 26, 30, 34, 38]

Test de la suite S6 : suite([2,4,6,10,16], [26,42,68,70,110,150,178], L).

L = [2, 4, 6, 10, 16, 26, 42, 68, 110, 178]

Lien du projet sur swish https://swish.swi-prolog.org/p/projet_prolog1.swinb#