

Exposé

PRÉSENTÉ PAR

- IMEN BENZAZA

- BOUMESSAOUD ABDELKADER

THEME

**ALGORITHME DE CHAINAGE AVANT EN
INTELLIGENCE ARTIFICIELLE (IA)**

Sous la direction de :

Monsieur Moussa BENAÏSSA

Résumé

Le projet consiste à développer un algorithme qui gère le chainage avant. Il est utilisé en intelligence artificielle, dans un système expert à base de règles, dans un moteur de règles, ou encore dans un système de production.

Structure de données choisies

Une règle : de type Règle, une classe à deux attributs

-Gauche : tableau dynamique contenant des chaînes de caractères qui caractérisent les éléments gauches d'une règle

-Droite : tableau dynamique contenant des chaînes de caractères qui caractérisent les éléments droites d'une règle

La base des faits : tableau dynamique contenant des chaînes de caractères qui caractérisent les éléments de la base des faits

La base des règles : tableau dynamique contenant des objets de type Règle qui caractérisent les règles de la base des règles

Pseudo-Code

Classe Regle

```
class Regle
{
|
|
|  String Gauche [];
|  String Droite [];
|
|  Regle() { }
|
}
```

Fonction siElementDansBF

Entrée : élément à vérifier s'il est dans la base des faits + la base des faits en elle-même

Sortie : (vrai/faux) si l'élément est dans la base des faits

Fonction siElementDansBF(BF[]: string ; element : string) : bool

```
{  
    existe : bool;  
    existe -> faux;  
    pour i de 0 jusqu'a taille de BF faire  
    {  
        si (element = BF[i])  
        {  
            existe -> vrai;  
        }  
    }  
    retourner existe;  
}
```

Function siDesElementsDansBF

Entrée : règle à vérifier si elle est vrai (tout ses gauches sont dans la base des fait) + la base des faits en elle-même

Sortie : (vrai/faux) si la règle est vrai

```
fonction siDesElementsDansBF(regle : regle ; BF[]: string):bool
{
    existent -> bool;
    i : integer
    i -> 0;
    faire
    {
        existent -> siElementDansBF (BF,regle.Gauche[i]);
        i++;
    }tant que ( existent = vrai && i < regle.Gauche.size());

    retourner existent;
}
```

Le Main

```
main
{
    taille_base_des_faits , nbr_regle , nbr_gauches , nbr_droites : integer ;
    i , j : integer ;
    temp : Regle ;
    maj : bool;
    maj -> faux;
    base_des_faits[] , element : string;
    base_des_regles : Regle;
    ecrire ("Combien d'elements y a t'il dans la base des faits ? : ");
    lire(taille_base_des_faits);
    Scan.nextLine();
```

```

pour i de 0 jusqu'a taille_base_des_faits faire
{
    ecrire("Donnez l'elements de la base des faits numero : ");
    ecrire (i+1)
    lire (element)
    ajouter a base_des_faits (element);
}

    ecrire("Nombre de regles : ");
lire (nbr_regle);

pour i de 0 jusqu'a nbr_regle faire
{
    temp -> null;
    ecrire("<-----regle : ");
    ecrire(i+1);
    ecrire("----->");
    ecrire("Nombre de gauches dans la regle : ");
    ecrire(i+1);
    lire (nbr_gauches);
    ecrire("Nombre de droites dans la regle : ");
    ecrire(i+1);
    lire(nbr_droites)
    pour j de 0 jusqu'a nbr_gauches faire
    {
        ecrire("Regle ");
        ecrire (i+1);
        ecrire(" Gauche ");
        ecrire(j+1);
        lire (element);
        temp.Gauche[j] -> element;
    }
    pour j de 0 jusqu'a nbr_droites faire
    {
        ecrire("Regle ");
        ecrire (i+1);
        ecrire(" Droite ");
        ecrire(j+1);
        lire (element);
        temp.Droite[j] -> element;
    }
    ajouter a base_des_regles (temp);
}

```

}

but : string;

ecrire("Donnez le but : ");

lire (but);

faire

{

pour i de 0 jusqu'a taille (base_des_regles) faire

{

si (siDesElementsDansBF(base_des_regles[i], base_des_faits))

{

maj -> true;

pour j de 0 jusqu'a taille (base_des_regles[i].Droite) faire

ajouter a base_des_faits (base_des_regles.[i].Droite.[j]);

}

sinon

maj -> false ;

}

}tant que (non(siElementDansBF(base_des_faits, but)) et maj = true);

si (siElementDansBF(base_des_faits,but))

ecrire("Reussite");

sinon

ecrire("Fail");

}

Caractéristiques du chainage choisis

chainage avant :

On commence avec les données disponibles (base des faits initiaux) et utilise des règles (base des règles) pour extraire davantage de données jusqu'à ce qu'un but soit atteint.

L'algorithme de chaînage avant effectue une recherche dans les règles jusqu'à ce qu'il trouve tous les gauches d'une règle vraie (dans la base des faits).

Une fois trouvé, il peut en déduire que la droite est vraie aussi, ce qui entraîne l'ajout de nouvelles informations à la base des faits.

Irrévocable :

Car les choix pris (règles appliquées) ne sont jamais remis en cause.

En profondeur :

Dans cet algorithme, lorsqu'une règle est déclenchée, les droites de la règle sont immédiatement rangées dans la base de faits.

L'arbre des solutions est exploré dans sa profondeur.

Code source de l'algorithme écrit en Java

Classe Regle

```
1 package com.company;
2
3 import java.util.ArrayList;
4
5 public class Regle
6 {
7     public ArrayList<String> Gauche = new ArrayList<String>(); //la partie gauche de la regle
8     public ArrayList<String> Droite = new ArrayList<String>(); //la partie droite de la regle
9
10    public Regle() { }
11    //getters pour utiliser les valeurs
12    public ArrayList<String> getGauche() { return Gauche; }
15
16    public ArrayList<String> getDroite() { return Droite; }
19
20 }
```

fonction siElementDansBF

```
1 package com.company;
2 import java.util.ArrayList;
3 import java.util.Scanner;
4
5 public class Main
6 {
7
8     @
9     public static boolean siElementDansBF(ArrayList<String> BF, String element) //verifie si un element est dans la base des faits
10    {
11        boolean existe = false; //si l'element existe dans la bf
12        for (int i = 0; i < BF.size(); i++)
13        {
14            if (element == BF.get(i))
15            {
16                existe = true;
17            }
18        }
19        return existe;
20    }
```

Fonction siDesElementsDansBF

```
21 @ public static boolean siDesElementsDansBF(Regle regle, ArrayList<String> BF) //verifie si les elements gauche d'une regle sont dans la base des faits
22 {
23     boolean existent ; //si plusieurs elements existent dans la bf
24     int i=0;
25     do
26     {
27         existent = siFlementDansBF (BF, regle.Gauche.get(i));
28         i++;
29     }while ( existent == true && i < regle.Gauche.size());
30
31     return existent;
32 }
33
34
```

Le Main

Partie déclaration des variables

```
35 ▶ public static void main(String[] args)
36 {
37     int taille_base_des_faits , nbr_regle , nbr_gauches , nbr_droites ; // la nomination est claire...
38     int i , j ; // variables d'indexation
39     Regle temp = new Regle() ; //regle intermediaire pour le remplissage de la base des regles
40     boolean maj = false; //variable de verification si la bf a change (mis a jour)
41     ArrayList<String> base_des_faits=new ArrayList<>();
42     ArrayList<Regle> base_des_regles=new ArrayList<>();
43
44     Scanner Scan = new Scanner(System.in);
```

Partie lecture de la base des faits

```
48 //remplissage de la base des faits
49 System.out.println("Combien d'elements y a t'il dans la base des faits ? : ");
50 taille_base_des_faits = Scan.nextInt();
51 Scan.nextLine();
52
53 for ( i = 0 ; i < taille_base_des_faits ; i++ )//boucle de remplissage de la bf
54 {
55     System.out.println("Donnez l'elements de la base des faits numero : "+(i+1)+" : ");
56     base_des_faits.add(Scan.nextLine());
57 }
```

Partie lecture de la base des regles

```
60 //remplissage des regles
61 System.out.println("Nombre de regles : ");
62 nbr_regle = Scan.nextInt();
63
64 for( i = 0 ; i < nbr_regle ; i++)//boucle de remplissage des regles
65 {
66     temp = null;
67
68     System.out.println("<-----regle : " +(i+1)+"----->");
69
70     System.out.println("Nombre de gauches dans la regle : " +(i+1)+" : ");
71     nbr_gauches = Scan.nextInt();
72
73     System.out.println("Nombre de droites dans la regle : " +(i+1)+" : ");
74     nbr_droites = Scan.nextInt();
75
76     for(j=0;j<nbr_gauches;j++) //remplissage de la partie gauche d'une regle
77     {
78         System.out.println("Regle "+(i+1)+" Gauche "+(j+1)+" : ");
79         temp.Gauche.set(j ,Scan.nextLine() );
80     }
81
82     for(j=0;j<nbr_droites;j++)//remplissage de la partie droite d'une regle
83     {
84         System.out.println("Regle "+(i+1)+" Droite "+(j+1)+" : ");
85         temp.Droite.set(j ,Scan.nextLine() );
86     }
87
88     base_des_regles.add(temp); //ajoute de la derniere regles input dans la bf
89
90 }
```

Partie traitement du chainage avant

```
96 //traitement chainage avant
97 do
98 {
99     for (i = 0; i < base_des_regles.size(); i++) //boucle de traitement de tt les regles
100     {
101         if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits)) //si tt les gauches d'une regles sont dans la bf
102         {
103             maj = true; // une maj a ete faite a la bf
104             for (j = 0; j < base_des_regles.get(i).Droite.size(); j++) //ajout de tt les droites d'une regles dans la bf
105                 base_des_faits.add(base_des_regles.get(i).Droite.get(j));
106         }
107         else
108             maj = false; // aucune maj a ete faite
109     }
110 }while (!siElementDansBF(base_des_faits, but)) && maj == true; //le but n'est pas dans la bf
111 // et une maj a ete faite a la bf pour continuer la chainage
```

Partie recherche du but dans la base des faits et affichage du résultat

```
113 //verification si le but est dans la bf
114 if ( siElementDansBF( base_des_faits, but ) )
115     System.out.println("Reussite");
116 else
117     System.out.println("Fail");
118 }
119 }
```

Exemple

Exemple1-a: Chaînage avant

Base de connaissances:

$BF_0 = \{A, D, E, G\}$

Base de règles (BR) : {

R1: A, B, C \Rightarrow H

R2: A, U, C \Rightarrow F

R3: E, G, B \Rightarrow S

R4: D, G \Rightarrow C

R5: A, E \Rightarrow B

R6: U, S, T \Rightarrow F

R7: G, H \Rightarrow R

R8: D, E \Rightarrow T

R9: R, S, H \Rightarrow F

R10: A, U \Rightarrow B }

Question :

En utilisant le **Chaînage avant**
en **profondeur d'abord**
monotone avec régime
irrévocable.

Montrer le **but F** ?

Teste du code

Lecture de la base des faits

```
Run: C:\Users\Surface\jdk\openjdk-16.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.2\lib\idea_rt.jar=62954:C:\Program Files\JetBra
Combin d'elements y a t'il dans la base des faits ? :
Donnez l'elements de la base des faits numero : 1 :
Donnez l'elements de la base des faits numero : 2 :
Donnez l'elements de la base des faits numero : 3 :
Donnez l'elements de la base des faits numero : 4 :
```

Lecture des règles

Règle 1

```
Nombre de regles :
10
<-----regle : 1----->
Nombre de gauches dans la regle : 1 :
3
Nombre de droites dans la regle : 1 :
1
Regle 1 Gauche 1 :
a
Regle 1 Gauche 2 :
b
Regle 1 Gauche 3 :
c
Regle 1 Droite 1 :
h
```

Règles 2 & 3

```
<-----regle : 2----->
Nombre de gauches dans la regle : 2 :
3
Nombre de droites dans la regle : 2 :
1
Regle 2 Gauche 1 :
a
Regle 2 Gauche 2 :
u
Regle 2 Gauche 3 :
c
Regle 2 Droite 1 :
f
<-----regle : 3----->
Nombre de gauches dans la regle : 3 :
3
Nombre de droites dans la regle : 3 :
1
Regle 3 Gauche 1 :
e
Regle 3 Gauche 2 :
g
Regle 3 Gauche 3 :
b
Regle 3 Droite 1 :
s
```


Règles 4 & 5

```
<-----regle : 4----->
Nombre de gauches dans la regle : 4 :
2
Nombre de droites dans la regle : 4 :
1
Regle 4 Gauche 1 :
d
Regle 4 Gauche 2 :
g
Regle 4 Droite 1 :
c
<-----regle : 5----->
Nombre de gauches dans la regle : 5 :
2
Nombre de droites dans la regle : 5 :
1
Regle 5 Gauche 1 :
a
Regle 5 Gauche 2 :
e
Regle 5 Droite 1 :
b
```

Règles 6 & 7

```
<-----regle : 6----->
Nombre de gauches dans la regle : 6 :
3
Nombre de droites dans la regle : 6 :
1
Regle 6 Gauche 1 :
u
Regle 6 Gauche 2 :
s
Regle 6 Gauche 3 :
t
Regle 6 Droite 1 :
f
<-----regle : 7----->
Nombre de gauches dans la regle : 7 :
2
Nombre de droites dans la regle : 7 :
1
Regle 7 Gauche 1 :
g
Regle 7 Gauche 2 :
h
Regle 7 Droite 1 :
r
```

Règles 8 & 9

```
<-----regle : 8----->
Nombre de gauches dans la regle : 8 :
2
Nombre de droites dans la regle : 8 :
1
Regle 8 Gauche 1 :
d
Regle 8 Gauche 2 :
e
Regle 8 Droite 1 :
t
<-----regle : 9----->
Nombre de gauches dans la regle : 9 :
3
Nombre de droites dans la regle : 9 :
1
Regle 9 Gauche 1 :
r
Regle 9 Gauche 2 :
s
Regle 9 Gauche 3 :
h
Regle 9 Droite 1 :
f
```

Règles 10

```
<-----regle : 10----->
Nombre de gauches dans la regle : 10 :
2
Nombre de droites dans la regle : 10 :
1
Regle 10 Gauche 1 :
a
Regle 10 Gauche 2 :
u
Regle 10 Droite 1 :
b
Donnez le but :
f
Reussite

Process finished with exit code 0
```

Affichage du résultat : REUSSITE