



قسم الإعلام الآلي Département d'Informatique

Exposé

PRÉSENTÉ PAR - IMEN BENZAZA - BOUMESSAOUD ABDELKADER

THEME

ALGORITHME DE CHAINAGE AVANT EN INTELLIGENCE ARTIFICIELLE (IA)

Sous la direction de :

Monsieur Moussa BENAISSA

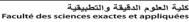


كلية العلوم الدقيقة والتطبيقية Faculté des sciences exactes et appliquées



<u>Résumé</u>

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 donnes choisis

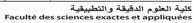
Une règle : de type Regle, une classe à deux attributs

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

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

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

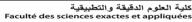
La base des regles : tableau dynamique contenant des objets de type Regle qui caractérisent les règles de la base des règles





Pseudo-Code

Classe 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

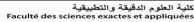
```
fonction siDesElementsDansBF(regle : regle ; BF[]: string):bool
{
    existent -> bool;
    i : integer
    i -> 0;
    faire
    {
        existent -> siElementDansBF (BF,regle.Gauche.[i]);
        i++;
    }tant que ( existent = vrai && i < regle.Gauche.size());
    retourner existent;
}</pre>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

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();
```

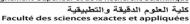


```
pour i de 0 jusqu'a taille_base_des_faits faire
  ecrire("Donnez l'elements de la base des faits numero : ");
 ecrire (i+1)
  lire (element)
 ajouter a base_des_faits (element);
    ecrire("Nombre de regles : ");
lire (nbr_regle);
pour i de 0 jusqu'a nbr_regle faire
  temp -> null;
  ecrire("<----regle:");
  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 règles vrai (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 fait.

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

```
package com.company;

import java.util.ArrayList;

public class Regle
{
    public ArrayList<String> Gauche = new ArrayList<String>();//la partie gauche de la regle
    public ArrayList<String> Droite = new ArrayList<String>();//la partie droite de la regle

public Regle() {}
    //getters pour utiliser les valeurs
    public ArrayList<String> getGauche() { return Gauche; }

public ArrayList<String> getDroite() { return Droite; }
}
```

fonction siElementDansBF

```
package com.company;
import java.util.ArrayList;
import java.util.Scanner;

public class Main
{

public static boolean siElementDansBF(ArrayList<String> BF, String element)//verifie si un element est dans la base des faits
}

boolean existe = false;//si l'element existe dans la bf
fon (int i = 0; i < BF.size(); i++)

{

if (element == BF.get(i))
 {

    existe = true;
 }

}

return existe;

}

return existe;
```



Fonction siDesElementsDansBF

```
public static boolean siDesFlementsDansBF(Regle regle,ArrayList<String> BF)//verifie si les elements gauche d'une règle sont dans la base des faits

boolean existent ;//si plusieurs elements existent dans la bf
int i=0;
do

{
    existent = siElementDansBF (BF,regle.Gauche.get(i));
    i++;
}
while (existent == true && i < regle.Gauche.size());
return existent;
}
```

Le Main

Partie déclaration des variables

```
public static void main(String[] args)

{
    int taille_base_des_faits , nbr_regle , nbr_gauches , nbr_droites ; // la nomination est claire...

int i , j ; // variables d'indexation

Regle temp = new Regle() ; //regle intermediaire poiur le remplissage de la base des regles

boolean maj = false; // variable de verification si la bf a change (mis a jour)

ArrayList<String> base_des_faits=new ArrayList<>();

ArrayList<Regle> base_des_regles=new ArrayList<>();

Scanner Scan = new Scanner(System.in);
```



Partie lecture de la base des faits

```
//remplissage de la base des faits
System.out.println("Combien d'elements y a t'il dans la base des faits ? : ");
taille_base_des_faits = Scan.nextInt();
Scan.nextLine();

for ( i = 0 ; i < taille_base_des_faits ; i++ )//boucle de remplissage de la bf

System.out.println("Donnez l'elements de la base des faits numero : "+(i+1)+" : ");
base_des_faits.add(Scan.nextLine());
}
```

Partie lecture de la base des regles



Partie traitement du chainage avant

```
//traitment chainage avant

do

{
    for (i = 0; i < base_des_regles.size(); i++)//boucle de traintemet de tt les regles

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))//si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))//si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), base_des_faits))/si tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), broite.size(); j++) //ajout de tt les droites d'une regles dans la bf

base_des_faits.add(base_des_regles.get(i), broite.size(); j++) //ajout de tt les droites d'une regles dans la bf

base_des_faits.add(base_des_regles.get(i), broite.size(); j++) //ajout de tt les gauches d'une regles sont dans le bf

if (siDesElementsDansBF(base_des_regles.get(i), broite.size(); j++) //ajout de tt les droites d'une regles dans la bf

base_des_faits.add(base_des_regles.get(i), broite.size(); j++) //ajout de tt les droites d'une regles dans la bf

base_des_faits.add(base_des_faits, but)) && maj == true); //le but n'est pas dans la bf

// et une maj a ete faite a la bf for (i) selles d'une regles dans la bf

// et une maj a ete faite a la bf for (i) s
```

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



Exemple

Exemple1-a: Chaînage avant

Base de connaissances:

```
BF<sub>0</sub> = {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 <u>Chaînage avant</u> en profondeur d'abord monotone avec régime irrévocable.

Montrer le **but F?**

Ghalem Belalem 18



Teste du code

Lecture de la base des faits

```
Runz

A c:\Users\Surface\.jdks\openjdk-16.8.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.2\lib\idea_rt.jar=62954:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1.2\lib\idea_rt.jar=6
```

Lecture des règles

Règle 1

```
Nombre de regles :

10

<----->
Nombre de gauches dans la regle : 1 :

3

Nombre de droites dans la regle : 1 :

1

Regle 1 Gauche 1 :

8

Regle 1 Gauche 2 :

B

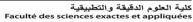
Regle 1 Droite 1 :

6
```



Règles 2 & 3

```
<---->
Nombre de gauches dans la regle : 2 :
Nombre de droites dans la regle : 2 :
Regle 2 Gauche 1 :
Regle 2 Gauche 2 :
Regle 2 Gauche 3 :
Regle 2 Droite 1 :
<---->
Nombre de gauches dans la regle : 3 :
Nombre de droites dans la regle : 3 :
Regle 3 Gauche 1 :
Regle 3 Gauche 2 :
Regle 3 Gauche 3 :
Regle 3 Droite 1 :
```





Règles 4 & 5

```
Nombre de gauches dans la regle : 4 :

Nombre de droites dans la regle : 4 :

Regle 4 Gauche 1 :

Regle 4 Gauche 2 :

Regle 4 Droite 1 :

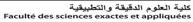
Nombre de gauches dans la regle : 5 :

Nombre de droites dans la regle : 5 :

Regle 5 Gauche 2 :

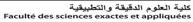
Regle 5 Groite 1 :

Regle 5 Droite 1 :
```





Règles 6 & 7





Règles 8 & 9

```
Nombre de gauches dans la regle : 8 :

Nombre de droites dans la regle : 8 :

Regle 8 Gauche 1 :

Regle 8 Gauche 2 :

Regle 8 Droite 1 :

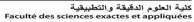
Nombre de gauches dans la regle : 9 :

Nombre de droites dans la regle : 9 :

Regle 9 Gauche 2 :

Regle 9 Gauche 3 :

Regle 9 Droite 1 :
```





Règles 10

```
Nombre de gauches dans la regle : 10 :

Nombre de droites dans la regle : 10 :

Regle 10 Gauche 1 :

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