

LO21

Marius Bozane
Louis Esteban

Automne 2020

Résumé

Pour notre projet de LO21 nous devons réaliser un système expert composé de bases de connaissances, de règles et d'un moteur d'inférence.

Une règle est composé d'une prémisse et d'une conclusion

Les règles sont sous la forme $A \wedge B \wedge C \Rightarrow D$

Avec $A \wedge B \wedge C$ comme prémisse et D comme conclusion.

Une base de connaissances est un ensemble de règles.

Le moteur d'inférence permet de déduire des faits certains en partant de la base de faits et en appliquant les règles de la base de connaissances

Table des matières

I	Choix de conception et d'implémentation	3
1	Choix de conception	4
2	Choix d'implémentation	4
II	Algorithmes	5
3	Règles	6
3.1	Nouvelle Règle	6
3.2	Ajout de Prémisses	6
3.3	Créer une conclusion	7
3.4	Test 1 : Une prémissse appartient à une règle	8
3.5	Supprimer une prémissse d'une règle	9
3.6	Test 2 : Prémissse vide d'une règle	10
3.7	Accéder à la première prémissse d'une règle	10
3.8	Accéder à la conclusion d'une règle	11
4	Base de Connaissances	12
4.1	Nouvelle Base de Connaissances	12
4.2	Ajouter une nouvelle règle	12
4.3	Afficher la première règle	13
5	Moteur d'Inférence	14
5.1	Moteur d'Inférence	14
5.2	Règle Vraie	14
5.3	Prémissse Vraie	16

5.4	Ajouter un fait à une liste	16
III	Jeux d’essais	17
IV	Commentaires sur les résultats	19

Première partie

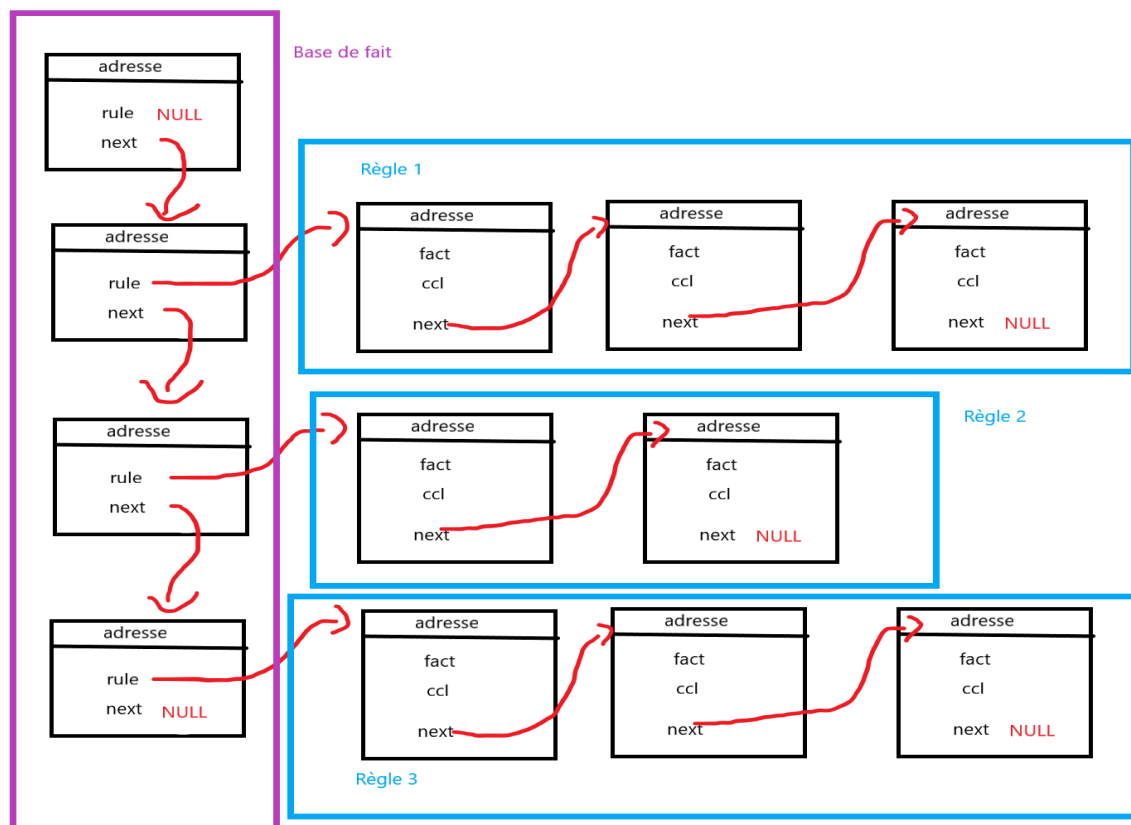
Choix de conception et d'implémentation

1 Choix de conception

Pour réaliser notre projet nous avons décidé de créer une structure pour les règles, une structure pour les bases de connaissances ainsi qu'une structure pour le moteur d'inférence.

Nous avons séparé le projet en 3 fichiers .c et 3 fichiers .h, chacun correspondant au programme et à la librairie associé de respectivement les règles, les bases de connaissances et le moteur d'inférence.

Les structures des base de faits et des règles se présentent sous la forme ci-dessous :



2 Choix d'implémentation

Deuxième partie

Algorithmes

3 Règles

3.1 Nouvelle Règle

L'algorithme ci-dessous permet de créer une règle vide et de retourner son pointeur.

Algorithme 1 : RègleVide

Variables : R : La nouvelle Règle
Résultat : R : Règle
1 **Début** RègleVide()
2 $R \leftarrow règle_vide$
3 **Fin**

3.2 Ajout de Prémisses

L'algorithme ci-dessous permet d'ajouter une prémisses à une règle.

Algorithme 2 : AjoutPrémisse

Variables :
- P : Prémisses à rajouter
- R : Règle dont on veut rajouter une prémisses
- R' : Stockage du dernier objet d'une règle
- T : Stockage de l'avant-dernier objet de la règle
- NP : L'adresse de la nouvelle prémisses
Données : R : Règle
Résultat : R : Règle
1 **Début** AjoutPrémisse (C, R)
2 **Tant que** $Suivant(R) \neq indéfini$ **faire**
3 $T \leftarrow Suivant(R)$
4 $R \leftarrow Suivant(R)$
5 **Fin**
6 $Conclusion(R') \leftarrow R$
7 $Fait(NP) \leftarrow R$
8 **Si** $Conclusion(R')=1$ **alors**
9 $Suivant(T) \leftarrow NP$
10 $Suivant(NP) \leftarrow R'$
11 **FinSi**
12 **Sinon**
13 $Suivant(R) \leftarrow NP$
14 **FinSi**
15 **Fin**

3.3 Créer une conclusion

L'algorithme ci-dessous permet d'ajouter une conclusion à une règle.

Algorithme 3 : CréerConclusion

```
Variables :  
-  $C$ : Conclusion à rajouter  
-  $R$ : Règle dont on veut rajouter une conclusion  
-  $R'$ : Règle de transit  
Données :  $R$ : Règle  
Résultat :  $R$ : Règle  
1 Début CréerConclusion ( $C, R$ )  
2   Tant que  $Suivant(R) \neq \text{indéfini}$  faire  
3      $R \leftarrow Suivant(R)$   
4   Fin  
5   Si  $Conclusion(R) = 0$  alors  
6      $R' \leftarrow RègleVide()$   
7      $Fait(R') \leftarrow C$   
8      $Conclusion(R') \leftarrow 1$   
9      $Suivant(R) \leftarrow R'$   
10     $Résultat \leftarrow \text{Vrai}$   
11  FinSi  
12  Sinon  
13     $Résultat \leftarrow \text{Faux}$   
14  FinSi  
15 Fin
```


3.4 Test 1 : Une prémissse appartient à une règle

L'algorithme ci-dessous permet de vérifier si une règle contient une prémissse ou non, il retourne vrai si la prémissse a été trouvée et il retourne faux si cette prémissse n'est pas contenue dans la règle.

Algorithme 4 : TestPrémisse

```
Variables :  
-  $R$ : Règle  
-  $P$ : Prémissse que l'on veut tester  
Données :  $R$ : Règle  
Résultat :  $Resultat$ : Booléen  
1 Début TestPrémisse ( $P, R$ )  
2   Tant que  $Suivant(R) \neq \text{indéfini}$  faire  
3     Si  $ComparerCaractère(Fact(R), P) = 0$  alors  
4        $Résultat \leftarrow \text{Vrai}$   
5     FinSi  
6      $R \leftarrow Suivant(R)$   
7   Fin  
8   Si  $(Conclusion(R) = 0)$  ( et  $ComparerCaractère(Fact(R), P) = 0$  ) alors  
9      $Résultat \leftarrow \text{Vrai}$   
10  FinSi  
11  Sinon  
12     $Résultat \leftarrow \text{Faux}$   
13  FinSi  
14 Fin
```

3.5 Supprimer une prémissse d'une règle

L'algorithme ci-dessous permet de supprimer une prémissse d'une règle, il retourne Vrai si l'opération a pu être réalisé et retourne faux s'il n'a pas pu trouver la prémissse en question.

Algorithme 5 : SupprimerPrémisse

Variables :

- P : Prémisse à rajouter
- R : Règle dont on veut rajouter une prémisse
- R' : Stockage du dernier objet d'une règle
- T : Stockage de l'avant-dernier objet de la règle

Données : R : Règle

Résultat : *Resultat*: Booléen

```
1 Début SupprimerPrémisse ( $P, R$ )
2   Si PrémisseVide( $R=1$ ) alors
3     | Résultat  $\leftarrow$  Faux
4   FinSi
5   Sinon
6     | Tant que  $R \neq$  indéfini faire
7       |    $T \leftarrow R$ 
8       |   Si (ComparerCaractère(Fait( $R$ ),  $P$ )=1) ( et Conclusion( $R$ )=1) alors
9         |     Suivant( $T$ )  $\leftarrow$  Suivant( $R$ )
10        |     Libérer( $R$ )
11        |     Résultat  $\leftarrow$  Vrai
12        |   FinSi
13        |    $R \leftarrow$  Suivant( $R$ )
14      Fin
15      Résultat  $\leftarrow$  Faux
16    FinSi
17 Fin
```

3.6 Test 2 : Prémisse vide d'une règle

L'algorithme ci-dessous test si une règle contient des prémisses ou non. Il retourne Vrai si la règle ne contient pas de prémisses (elle peut tout de même contenir une conclusion) et retourne faux si elle contient des prémisses.

Algorithme 6 : PrémisseVide	
Variables : R : Règle que l'on veut tester	
Données : R : Règle	
Résultat : $Resultat$: Booléen	
1	Début PrémisseVide (R)
2	Si $Suivant(R) \neq indéfini$ alors
3	Résultat \leftarrow Faux
4	FinSi
5	Sinon si $Conclusion(Suivant(R))=I$ alors
6	Résultat \leftarrow Vrai
7	FinSi
8	Sinon
9	Résultat \leftarrow Faux
10	FinSi
11	Fin

3.7 Accéder à la première prémisse d'une règle

L'algorithme ci-dessous permet de connaître la première prémisse d'une règle (si celle ci en contient bien sûr). Ce dernier retourne un pointeur lié à la première prémisse de la règle.

Algorithme 7 : PremièrePrémisse	
Variables : R : Règle dont on veut voir la prémisse	
Données : R : Règle	
Résultat : P : Prémisse	
1	Début PremièrePrémisse (R)
2	Tant que $Suivant(R) \neq indéfini$ faire
3	$R \leftarrow Suivant(R)$
4	Fin
5	Si $Conclusion(R)=I$ alors
6	Retourner ($Fait(R)$)
7	FinSi
8	Retourner indéfini
9	Fin

3.8 Accéder à la conclusion d'une règle

L'algorithme ci-dessous permet de connaître la conclusion d'une règle (si celle ci existe bien sûr). Ce dernier retourne un pointeur lié à la conclusion de la règle.

Algorithme 8 : VoirConclusion	
	Variables : R : Règle dont on veut la conclusion
	Données : R : Règle
	Résultat : P : Prémisse
1	Début VoirConclusion (C, R)
2	Tant que $Suivant(R) \neq \text{indéfini}$ faire
3	$R \leftarrow Suivant(R)$
4	Fin
5	Si $Conclusion(Rule)=1$ alors
6	$P \leftarrow \text{Fait}(R)$
7	FinSi
8	Sinon
9	$P \leftarrow \text{indéfini}$
10	FinSi
11	Fin

4 Base de Connaissances

4.1 Nouvelle Base de Connaissances

L'algorithme ci-dessous permet de créer une base de connaissances vide. Il retourne le pointeur lié à cette base de connaissances.

Algorithme 9 : BC_Vide

<p>Variables : <i>BC</i>: La nouvelle Base de Connaissances Résultat : <i>BC</i>: Base de Connaissances</p> <pre>1 Début BC_Vide() 2 <i>BC</i> ← <i>BC_vide</i> 3 Fin</pre>

4.2 Ajouter une nouvelle règle

L'algorithme ci-dessous permet d'ajouter à une base de connaissances existante une nouvelle règle. Il retourne vrai si l'opération s'est déroulé avec succès.

Algorithme 10 : AjoutRègle

<p>Variables : - <i>R</i>: Règle à rajouter - <i>BC</i>: Base dont on veut rajouter une règle - <i>NR</i>: Nouvelle règle Données : <i>BC</i>: Base de Connaissances Résultat : <i>Resultat</i>: Booléen</p> <pre>1 Début AjoutRègle (<i>R</i>,<i>BC</i>) 2 <i>NR</i> ← <i>BC_Vide</i>() 3 Tant que <i>Suivant</i>(<i>BC</i>) ≠ indéfini faire 4 <i>BC</i> ← <i>Suivant</i>(<i>BC</i>) 5 Fin 6 <i>Suivant</i>(<i>BC</i>) ← <i>NR</i> 7 <i>Règle</i>(<i>NR</i>) ← <i>R</i> 8 Retourner Vrai 9 Fin</pre>

4.3 Afficher la première règle

L'algorithme ci-dessous permet d'afficher la première règle d'une base de connaissance. Il retourne le pointeur lié à la première règle de la base (si celle-ci existe).

Algorithme 11 : PremièreRègle	
Variables : BC : La base de connaissances en question	
Résultat : R : Règle	
1	Début PremièreRègle(BC)
2	Si $Suivant(BC) = indéfini$ alors
3	Retourner indéfini
4	FinSi
5	Sinon
6	Retourner Règle($Suivant(BC)$)
7	FinSi
8	Fin

5 Moteur d'Inférence

5.1 Moteur d'Inférence

L'algorithme ci-dessous retourne une liste chaînée des faits vrais.

Algorithme 12 : MoteurInférence

Variables :

- LF : Liste de Faits
- BC : Base connaissances
- KB : Base de connaissances transitoire
- PF : Liste de faits transitoires
- RF : Liste de faits transitoire

Données : LF : Liste de faits

Résultat : R : Règle

```
1 Début MoteurInférence( $BC, LF$ )
2   Tant que  $PF \neq \text{indéfini}$  faire
3     |   AjouterFaitsListe( $PF, F$ )
4     |    $PF \leftarrow \text{Suivant}(PF)$ 
5   Fin
6   Tant que  $KB \neq \text{indéfini}$  faire
7     |   Si RègleVraie(Règle( $KB$ ),  $RF$ ) = Vrai alors
8     |     |   AjouterFaitsListe(VoirConclusion(Rule( $KB$ ),  $RF$ ))
9     |   FinSi
10    |    $KB \leftarrow \text{Suivant}(KB)$ 
11  Fin
12  Retourner  $RF$ 
13 Fin
```

5.2 Règle Vraie

L'algorithme ci-dessous permet de savoir si une règle est vraie. Il retourne indéfini si la règle est fausse et la conclusion si la règle est vraie

Algorithme 13 : RègleVraie	
	Variables :
	- LF : Une liste de faits
	- R : Une règle
	Données : LF : Une liste de faits R : Une règle
	Résultat : $Resultat$: Un fait
1	Début RègleVraie(BC)
2	Si ($PrémisseVide(R) = Faux$) (et $VoirConclusion(R) \neq indéfini$) alors
3	Tant que $Suivant(R) \neq indéfini$ faire
4	Si $PrémisseVraie(Fait(R), LF) = Faux$ alors
5	Retourner indéfini
6	FinSi
7	$R \leftarrow Suivant(R)$
8	Fin
9	Retourner $Fait(R)$
10	FinSi
11	Sinon
12	Retourner indéfini
13	FinSi
14	Fin

5.3 Prémisse Vraie

L'algorithme ci-dessous permet de tester si une prémisse est vraie. Il retourne vrai si la prémisse est vraie et faux si la prémisse est fausse.

Algorithme 14 : PrémisseVraie	
Variables : - LF : La liste de faits - F : Un fait Données : LF : Liste de faits Résultat : $Resultat$: Booléen	
1	Début PrémisseVraie(F, LF)
2	Tant que $Suivant(LF) \neq \text{indéfini}$ faire
3	Si $ComparerCaractère(Fait(LF), F) = 0$ alors
4	Retourner Vrai
5	FinSi
6	$LF \leftarrow Suivant(LF)$
7	Fin
8	Si $ComparerCaractère(Fait(LF), F) = 0$ alors
9	Retourner Vrai
10	FinSi
11	Sinon
12	Retourner Faux
13	FinSi
14	Fin

5.4 Ajouter un fait à une liste

L'algorithme ci-dessous

Algorithme 15 : PremièreRègle	
Variables : BC : La base de connaissances en question Résultat : R : Règle	
1	Début PremièreRègle(BC)
2	Si $Suivant(BC) = \text{indéfini}$ alors
3	Retourner indéfini
4	FinSi
5	Sinon
6	Retourner Règle($Suivant(BC)$)
7	FinSi
8	Fin

Troisième partie

Jeux d'essais

```

#include <string.h>
#include <stdio.h>
#include "kb.h"
#include "rule.h"
#include "ie.h"

int main(){
    char a[3] = "A";
    char b[3] = "B";
    char c[3] = "C";
    char d[3] = "D";
    rule_t *pregle;
    pregle = new_rule();

    printf("test 3: %d\n", test_empty_facts(pregle));
    add_fact(&a, pregle);
    add_fact(&b, pregle);
    printf("test 2 : %s\n", pregle->next->next->fact);

    create_ccl(&c, pregle);

    printf("test 3: %d\n", test_fact(&c, pregle));

    return 0;
}

```

```

D:\dev\git\LOP21_Project\LO21_Project\Lo21\bin\Debug\Lo21.exe
test 3: 1
test 5 : 1389328
test 2 : B
test 4 b : 0
test 4 b : 0
test 3: 0

Process returned 0 (0x0)   execution time : 0.014 s
Press any key to continue.

```

Quatrième partie

Commentaires sur les résultats