LO21

Marius Bozane
Louis Esteban

Automne 2020

Résumé

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

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 règle.

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

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.

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
Début RègleVide()
  $R \longleftarrow$ *règle_vide*
Fin

3.2 Ajout de Prémisses

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

Algorithme 2 : AjoutPrémisse

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
- NP: L'adresse de la nouvelle prémisse

Données : R: Règle
Résultat : R: Règle
Début AjoutPrémisse (C,R)
  Tant que *Suivant(*R*)* $\neq$ *indéfini* **faire**
    $T \longleftarrow$ Suivant(R)
    $R \longleftarrow$ Suivant(R)
  Fin
  Conclusion(R') $\longleftarrow$ R
  Fait(NP) $\longleftarrow$ R
  Si *Conclusion(*R'*)=1* **alors**
    Suivant(T) $\longleftarrow$ NP
    Suivant(NP) $\longleftarrow$ R'
  FinSi
  Sinon
    Suivant(R) $\longleftarrow$ NP
  FinSi
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) ≠ indéfini faire
3          R ⟵ Suivant(R)
4      Fin
5      Si Conclusion(R) = 0 alors
6          R' ⟵ RègleVide()
7          Fait(R') ⟵ C
8          Conclusion(R') ⟵ 1
9          Suivant(R) ⟵ R'
10         Résultat ⟵ Vrai
11     FinSi
12     Sinon
13         Résultat ⟵ Faux
14     FinSi
15 Fin
```

3.4 Test 1 : Une prémisse appartient à une règle

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

Algorithme 4 : TestPrémisse

Variables :
- R: Règle
- P: Prémisse que l'on veut tester

Données : R: Règle

Résultat : $Resultat$: Boléen

```
1  Début TestPrémisse (P,R)
2      Tant que Suivant(R) ≠ indéfini faire
3          Si ComparerCaractère(Fact(R),P)=0 alors
4              Résultat ⟵ Vrai
5          FinSi
6          R ⟵ Suivant(R)
7      Fin
8      Si ( et Conlusion(R)=0) (ComparerCaractère(Fact(R),P)=0) alors
9          Résultat ⟵ Vrai
10     FinSi
11     Sinon
12         Résultat ⟵ Faux
13     FinSi
14 Fin
```

3.5 Supprimer une prémisse d'une règle

L'algorithme ci-dessous permet de supprimer une prémisse d'une règle, il retourne Vrai si l'opération a pu être réalisé et retourne faux s'il n'a pas pu trouver la prémisse 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 : $Resultat$: Boléen

```
1  Début SupprimerPrémisse (P,R)
2      Si PrémisseVide(R=1 alors
3          Résultat ⟵ Faux
4      FinSi
5      Sinon
6          Tant que R ≠ indéfini faire
7              T ⟵ R
8              Si ( et ComparerCaractère(Fait(R),P)=1) (Conclusion(R)=1) alors
9                  Suivant(T) ⟵ Suivant(R)
10                 Libérer(R)
11                 Résultat ⟵ Vrai
12             FinSi
13             R ⟵ Suivant(R)
14         Fin
15         Résultat ⟵ 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 : Boléen
1  Début PrémisseVide (R)
2  |  Si Suivant(R) ≠ indéfini alors
3  |  |  Résultat ⟵ Faux
4  |  FinSi
5  |  Sinon si Conclusion(Suivant(R))=1 alors
6  |  |  Résultat ⟵ Vrai
7  |  FinSi
8  |  Sinon
9  |  |  Résultat ⟵ 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) ≠ indéfini faire
3  |  |  R ⟵ Suivant(R)
4  |  Fin
5  |  Si Conclusion(R)=1 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) ≠ indéfini faire
 3 |   |   R ⟵ Suivant(R)
 4 |   Fin
 5 |   Si Conclusion(Rule)=1 alors
 6 |   |   P ⟵ Fait(R)
 7 |   FinSi
 8 |   Sinon
 9 |   |   P ⟵ 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

Variables : BC: La nouvelle Base de Connaissances
Résultat : BC: Base de Connaissances
1 **Début** BC_Vide()
2 | $BC \longleftarrow$ *BC_vide*
3 **Fin**

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

Variables :
- R: Règle à rajouter
- BC: Base dont on veut rajouter une règle
- NR: Nouvelle règle

Données : BC: Base de Connaissances
Résultat : $Resultat$: Boléen
1 **Début** AjoutRègle (R,BC)
2 | $NR \longleftarrow$ *BC_Vide()*
3 | **Tant que** *Suivant*(BC) $\neq$ *indéfini* **faire**
4 | | $BC \longleftarrow$ Suivant(BC)
5 | **Fin**
6 | Suivant(BC) $\longleftarrow NR$
7 | Règle(NR) $\longleftarrow R$
8 | Retourner Vrai
9 **Fin**

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 prémisses vraies.

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

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.

Troisième partie

Commentaires sur les résultats