Chapitre 4 (suite)
Le Calcul de prédicats

But:

• Introduire la méthode de résolution pour le calcul de prédicats

Résolution par réfutation

Cas du calcul des propositions

1. Convertir les FBFs en une forme clausale (conjonction de clauses)

2 . Convertir la négation de la FBF, A, à prouver en une forme clausale

3. Combiner les clauses résultant des 2 étapes précédentes dans un ensemble W (*les clauses combinées sont appelées* ***clauses parentes)***

4. Appliquer itérativement la résolution pour les clauses dans W jusqu'à il n'y pas de résultat à ajouter ou la clause vide est produite

La clause résultante est appelée clause ***résolvante,***

Résolution

- **Pour le calcul des prédicats** :

1. Un littéral et sa négation s'éliminent s'il existe une substitution S qui unifie les deux littéraux. S est ensuite appliquée aux résultat de la résolution. S doit être l'unificateur le plus général.

2. Les substitutions utilisées dans la preuve par résolution constituent l'ensemble des littéraux/variables pour lesquels la clause à prouver est vraie.

Résolution

Forme clausale

1. **Éliminer les implications**

Utiliser l'équivalence $P \rightarrow Q \equiv \neg P \vee Q$ pour éliminer les implications

2. **Réduire la portée de la négation**

Utiliser les lois de DeMorgan

$\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$ et $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$

et d'autres transformations

$\neg(\neg P) \equiv P$

$\neg \exists X\ p(X) \equiv \forall X \neg p(X)$

$\neg \forall X\ p(X) \equiv \exists X \neg p(X)$

Résolution

3. Standardiser les variables

Renommer les variables de façon à avoir des variables différentes pour chaque quantificateur.

$$\forall X\ q(X) \vee \forall X\ q(X) \equiv \forall X\ q(X) \vee \forall Y\ q(Y)$$

4. **Mettre sous forme prenexe**

Déplacer toutes les quantificateurs au début de l'expression en gardant leur ordre d'apparition.

Résolution

5. **Éliminer les quantificateurs existentiels (skolémization)**

- Remplacer toutes les variables quantifiées existentiellement par un symbole de fonction dont les arguments sont les variables qui sont quantifiées universellement qui précèdent notre variable.
- Supprimer les quantificateurs existentiels devenus inutiles.

6. Éliminer les quantificateurs universels

7. Mettre la formules sous forme de conjonctions de disjonctions

Utiliser les propriétés de distributivité

$$(P \vee (Q \wedge R)) \equiv (P \vee Q) \wedge (P \vee R)$$

Résolution

8. Éliminer les conjonctions

On obtient un ensemble de clauses

9. Standardiser les variables

Renommer les variables de façon que chaque 2 clauses ont des variables différentes

Exemples

1. $person(Marcus) \Rightarrow person(Marcus)$
2. $pompeiian(Marcus) \Rightarrow pompeiian(Marcus)$
3. $\forall x\ (pompeiian(x) \rightarrow roman(x)) \Rightarrow \neg\ pompeiian(x) \vee roman(x)$
4. $ruler(Caesar) \Rightarrow ruler(Caesar)$
5. $\forall x\ (roman(x) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar)) \Rightarrow$
   $\neg\ roman(z) \vee loyalto(z,Caesar) \vee hate(z,Caesar)$
6. $\forall x \exists y\ loyalto(x,y) \Rightarrow loyalto(w, f(w))$
7. $\forall x\ \forall y\ (person(x) \wedge ruler(y) \wedge tryAssassinate(x,y) \rightarrow \neg\ loyalto(x,y)) \Rightarrow$
   $\neg\ person(u) \vee \neg\ ruler(y) \vee \neg\ tryAssassinate(u,y) \vee \neg\ loyalto(u,y)$
8. $tryAssassinate(Marcus,Caesar) \Rightarrow tryAssassinate(Marcus,Caesar)$

Règles de résolution pour les prédicats

- Étant donné 2 clauses $\cup_i \{ L_i \}$ et $\cup_j \{ M_j \}$, la clause résolvante est calculée comme suit:
- Trouver un littéral L_k et un littéral M_l pour lesquels il existe un unificateur le plus général θ tel que

$$\theta(L_k) = \neg\, \theta(M_l)$$

  - Alors la clause résolvante serait

$$\cup_i \{\theta(L_i)\} \setminus \{\theta(L_k)\} \cup_j \{\theta(M_j)\} \setminus \{\theta(M_l)\}$$

Factorisation

- *Soit C* une clause contenant 2 littéraux ayant un unificateur le plus général θ, la clause obtenue par $\theta(C)$ et après avoir éliminé les littéraux identiques est appelé facteur de C et elle est une conséquence logique de C.
- La Factorisation consiste à générer un facteur à partir d'une clause.
- La règle de résolution est sound mais pas complète.
- La résolution et la factorisation représente ensemble un système de preuve complet.

Preuve par Résolution

Pour prouver que la formule *f* implique la formule *g*

- Transformer *f* en un ensemble de clauses
- Transformer $\neg\, g$ en un autre ensemble de clauses
- Combiner les 2 ensembles de clauses pour obtenir un seul ensemble
- Appliquer répétitivement la règle de *résolution et la factorisation* à cet ensemble jusqu'à générer la clause vide, notée aussi [].

  À chaque étape appliquer la règle de résolution à 2 clauses de l'ensemble courant ou la factorisation à une clause de l'ensemble courant. Ajouter la clause résolvante résultat ou le facteur à l'ensemble courant pour obtenir un nouvel ensemble, qui devient l'ensemble courant de la prochaine étape. Arrêter quand l'ensemble contient la clause vide.

Exemple

Étant donné les faits, observations et règles suivantes

1. Marcus is a person
2. Marcus was a Pompeiian
3. All Pompeiians were Romans
4. Caesar was a ruler
5. All Romans are loyal to Caesar or hate him
6. Everyone is loyal to somebody.
7. The only rulers people try to assassinate are those to whom they are not loyal
8. Marcus tried to assassinate Caesar

Prouver que Marcus hated Caesar

Exemple

Formellement, étant donné la conjonction des formules suivantes:

1. *person(Marcus)*
2. *pompeiian(Marcus)*
3. $\forall x\ (pompeiian(x) \rightarrow roman(x))$
4. *ruler(Caesar)*
5. $\forall x\ (roman(x) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar))$
6. $\forall x \exists y\ loyalto(x,y)$
7. $\forall x\ \forall y\ (person(x) \wedge ruler(y) \wedge tryAssassinate(x,y) \rightarrow \neg\ loyalto(x,y))$
8. *tryAssassinate(Marcus,Caesar)*

Prouver que *hate(Marcus,Caesar)* *en* est une conséquence logique

Exemple

Nous prouvons que la conjonction des formules suivantes est inconsistante

1. *person(Marcus)*
2. *pompeiian(Marcus)*
3. $\forall x\ (pompeiian(x) \rightarrow roman(x))$
4. *ruler(Caesar)*
5. $\forall x\ (roman(x) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar))$
6. $\forall x \exists y\ loyalto(x,y)$
7. $\forall x\ \forall y\ (person(x) \wedge ruler(y) \wedge tryAssassinate(x,y) \rightarrow \neg\ loyalto(x,y))$
8. *tryAssassinate(Marcus,Caesar)*
9. $\neg$ ***hate(Marcus,Caesar)***

Exemple

Convertir les formules en un ensemble de clauses

1. *person(Marcus)*
2. *pompeiian(Marcus)*
3. $\neg$ *pompeiian(x)* $\vee$ *roman(x)*
4. *ruler(Caesar)*
5. $\neg$ *roman(z)* $\vee$ *loyalto(z,Caesar)* $\vee$ *hate(z,Caesar)*
6. *loyalto(w, f(w))*
7. $\neg$ *person(u)* $\vee$ $\neg$ *ruler(y)* $\vee$ $\neg$ *tryAssassinate(u,y)* $\vee$ $\neg$ *loyalto(u,y)*
8. *tryAssassinate(Marcus,Caesar)*
9. $\neg$ *hate(Marcus,Caesar)*

Si l'ensemble des clauses est inconsistant la preuve par résolution donnera la clause vide

Extraction de réponse

- Une trace de preuve par résolution peut être utiliser pour répondre à une question.

- L'idée est lorsqu'on prouve que *g* est une conséquence logique de *f*,
  - ajouter le littéral *answer*(x_1,…,x_n) à chaque clause de la forme clausale de *g*, avec x_1, …, x_n les variables apparaissant dans la clause.
  - ensuite appliquer la procédure de résolution, mais s'arrêter quand nous obtenons une clause composée des littéraux answer (au lieu de la clause vide).

Exemple

1. Marcus is a person
2. Marcus was a Pompeiian
3. All Pompeiians were Romans
4. Caesar was a ruler
5. All Romans are loyal to Caesar or hate him
6. The only rulers people try to assassinate are those to whom they are not loyal
7. Marcus tried to assassinate Caesar

Is there someone who hates Caesar ?

Exemple

Formellement, étant donné la conjonction des formules suivantes:

1. $person(Marcus)$
2. $pompeiian(Marcus)$
3. $\forall x\ (pompeiian(x) \rightarrow roman(x))$
4. $ruler(Caesar)$
5. $\forall x\ (roman(x) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar))$
6. $\forall x \exists y\ loyalto(x,y)$
7. $\forall x\ \forall y\ (person(x) \wedge ruler(y) \wedge tryAssassinate(x,y) \rightarrow \neg\ loyalto(x,y))$
8. $tryAssassinate(Marcus,Caesar)$

prouver que $\exists x\ hate(x,Caesar)$ *est une conséquence logique et donner la valeur de x.*

Exemple

Ajouter la négation de $\exists x\ hate(x,Caesar)$ aux autres formules:

1. $person(Marcus)$
2. $pompeiian(Marcus)$
3. $\forall x\ (pompeiian(x) \rightarrow roman(x))$
4. $ruler(Caesar)$
5. $\forall x\ (roman(x) \rightarrow loyalto(x,Caesar) \vee hate(x,Caesar))$
6. $\forall x \exists y\ loyalto(x,y)$
7. $\forall x\ \forall y\ (person(x) \wedge ruler(y) \wedge tryAssassinate(x,y) \rightarrow \neg\ loyalto(x,y))$
8. $tryAssassinate(Marcus,Caesar)$
9. $\neg\ \exists x\ hate(Marcus,Caesar)$ (equivalent to $\forall x \neg\ hate(Marcus,Caesar)$)

Exemple

Convertir la formule en forme clausale, ajouter le littéral *answer* aux clauses résultant de $\neg \exists x$ *hate(Marcus,Caesar),* puis appliquer la résolution jusqu'à l'obtention de la clause *answer* .

1. *person(Marcus)*
2. *pompeiian(Marcus)*
3. $\neg$ *pompeiian(x)* $\vee$ *roman(x)*
4. *ruler(Caesar)*
5. $\neg$ *roman(z)* $\vee$ *loyalto(z,Caesar)* $\vee$ *hate(z,Caesar)*
6. *loyalto(w, f(w))*
7. $\neg$ *person(u)* $\vee$ $\neg$ *ruler(y)* $\vee$ $\neg$ *tryAssassinate(u,y)* $\vee$ $\neg$ *loyalto(u,y)*
8. *tryAssassinate(Marcus,Caesar)*
9. $\neg$ *hate(v,Caesar)* $\vee$ *answer(v)*