Preuve par résolution

- Procédure générale pour faire de l’inférence
  - modus ponens et l’instantiation universelle sont des cas particuliers

- Cette procédure est correcte et complète (sous certaine condition, à voir plus tard)

- On aura besoin des outils suivants :
  - la **substitution**
  - l’**unification**
  - la **transformation sous forme normale conjonctive**

Hugo Larochelle et Froduald Kabanza

Preuve par résolution

- Pour prouver que f_1 implique f_2
  - transformer f_1 en un ensemble de clauses en forme normale conjonctive
  - y ajouter les clauses pour $\neg f_2$ (comme dans preuve par contradiction)
  - appliquer répétitivement la **règle de résolution** jusqu'à aboutir à la clause vide, notée $\square$ (on a prouvé que f_1 implique f_2)
  - s'il n'est plus possible d'appliquer la règle de résolution, f_1 n'implique pas f_2

- **Règle de résolution pour le cas propositionnel** (c.-à-d. prédicats sans arguments) :
  - étant données les **clauses parents** $(p_1 \vee \ldots \vee p_n)$ et $(\neg p_1 \vee q_1 \vee \ldots \vee q_m)$, on génère la **clause résolvante** $p_2 \vee \ldots p_n \vee q_1 \vee \ldots \vee q_m$
  - on retrouve la règle Modus Ponens, puisque $f_1 \rightarrow f_2$ est équivalent à $\neg f_1 \vee f_2$

Règle de résolution pour les prédicats

- **Règle de résolution pour le cas de prédicats généraux**

  Soit deux clauses parents $L = L_1 \vee \ldots \vee L_n$ et $M = M_1 \vee \ldots \vee M_m$:

  1. trouver un littéral L_k et un littéral M_l tel qu'il existe un UPG θ tel que $L_k\theta = \neg M_l\theta$
  2. la clause résolvante de $L_1 \vee \ldots \vee L_n$ et $M_1 \vee \ldots \vee M_m$ est

$$\underbrace{L_1\theta \vee \ldots \vee L_{k-1}\theta \vee L_{k+1}\theta \vee \ldots \vee L_n\theta}_{\boldsymbol{L\,\theta \text{ sans } L_k\theta}} \vee \underbrace{M_1\theta \vee \ldots \vee M_{l-1}\theta \vee M_{l+1}\theta \vee \ldots \vee M_m\theta}_{\boldsymbol{M\,\theta \text{ sans } M_l\theta}}$$

- Ex. :
  - **clauses parents:** $L = \neg dog(x) \vee animal(x)$, $M = \neg animal(y) \vee die(y)$
  - **clause résolvante:** $\neg dog(x) \vee die(x)$ (UPG = $\{ y = x \}$)
- Deux clauses parents peuvent avoir plusieurs résolvants selon le choix L_k et M_l

Règle de résolution pour les prédicats

- La règle de résolution telle que décrite jusqu'à maintenant est **correcte** (***sound***), mais elle n'est pas **complète**

- La règle de résolution combinée avec la **factorisation** est **complète**
  - si deux littéraux d'une même clause ont un UPG, on remplace ces littéraux par le résultat de leur unification
  - un **facteur** est le résultat de cette transformation
    - ex. : $q(z) \vee p(f(y))$ est un facteur de la clause $q(z) \vee p(x) \vee p(f(y))$ (obtenu par l'UPG $\{ x = f(y) \}$)
  - on peut utiliser la factorisation au besoin, avant ou après l'application de la règle de résolution, afin de générer de nouvelles clauses

Exemple 1

- Tous les chiens sont des animaux
  - $\forall\, x\ dog(x) \rightarrow animal(x)$
- Tous les animaux vont mourir
  - $\forall\, y\ animal(y) \rightarrow die(y)$
- Fido est un chien
  - $dog(Fido)$
- Prouvez que Fido va mourir
  - $die(Fido)$

Exemple 1

Formules

1. $\forall\, x\; dog(x) \rightarrow animal(x)$
2. $\forall\, y\; animal(y) \rightarrow die(y)$
3. $dog(Fido)$

Niez la conclusion que Fido va mourir

4. $\neg\; die(Fido)$

- **Forme normale conjonctive**

1. $\neg\; dog(x) \vee animal(x)$
2. $\neg\; animal(y) \vee die(y)$
3. $dog(Fido)$
4. $\neg\; die(Fido)$

---

5. $\neg\; dog(y) \vee die(y)$
   1, 2, $\{x=y\}$
6. $die(Fido)$
   3, 5 $\{y=Fido)$
7. $\square$
   4, 6

Exemple 2

1. Marcus est une personne.
2. Marcus est un pompéien.
3. Tous les pompéiens sont des romains.
4. César est un dirigeant.
5. Tout le monde est loyal à quelqu'un.
6. Tous les romains sont loyaux à César ou le haïssent.
7. Les seuls dirigeants qu'une personne essaie d'assassiner sont ceux auxquels elle n'est pas loyal
8. Marcus a essayer d'assassiner César.

Prouvez que Marcus hait César

1. $personne(Marcus)$
2. $pompeien(Marcus)$
3. $\forall$ x $pompeien(x) \rightarrow romain(x)$
4. $dirigeant(Cesar)$
5. $\forall x \exists y\ loyal(x,y)$
6. $\forall x\ romain(x) \rightarrow loyal(x,Cesar) \vee hait(x,Cesar)$
7. $\forall$ x $\forall$ y $personne(x) \wedge dirigeant(y) \wedge assassiner(x,y) \rightarrow \neg\ loyal(x,y)$

8. $assassiner(Marcus,Cesar)$

Prouvez : $hait(Marcus,Cesar)$

Etape 10 : Ajouter les clauses de la négation de l'expression à prouver

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ¬ *pompeien*(*x1*) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. *loyal*(*x2*, *f1*(*x2*))
6. ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨
   ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ¬ *pompeien*(*x1*) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. *loyal*(*x2*, *f1*(*x2*))
6. ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨
   ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)

9. ¬ *hait*(*Marcus*, *Cesar*)

Etape 11 : Appliquer la résolution itérativement jusqu'à la clause vide

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ¬ *pompeien*(*x1*) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. *loyal*(*x2*, *f1*(*x2*))
6. ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨ ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)
9. ¬ *hait*(*Marcus*, *Cesar*)
10. *romain*(*Marcus*) — 2, 3, {*x1*=*Marcus*}
11. *loyal*(*Marcus*,*Cesar*) ∨ *hait*(*Marcus*,*Cesar*) — 6, 10, {*x4*=*Marcus*}
12. *loyal*(*Marcus*,*Cesar*) — 9, 11
13. ¬ *personne*(*Marcus*) ∨ ¬ *dirigeant*(*Cesar*) ∨ ¬ *assassiner*(*Marcus*,*Cesar*) — 7, 12, {*x5*=*Marcus*, *x6*=*Cesar*}
14. ¬ *personne*(*Marcus*) ∨ ¬ *dirigeant*(*Cesar*) — 8,13
15. ¬ *personne*(*Marcus*) — 4,14
16. □ (clause vide) — 1,15

Hugo Larochelle et Froduald Kabanza

Répondre à des questions

- La résolution permet de savoir si oui ou non, f_2 est une conséquence logique de f_1
- On peut aussi exploiter les traces du processus de preuve pour trouver des valeurs des variables qui permettent de déduire que f_2 est une conséquence logique de f_1 :
  - on ajoute $Rep(x_1, \ldots, x_n)$ à chaque clause de $\neg f_2$, où les x_i sont les variables apparaissant dans la clause
  - on applique la preuve par résolution
  - on arrête lorsqu'on a une clause composée uniquement du littéral *Rep*

Hugo Larochelle et Froduald Kabanza

Exemple 3. Répondre à la question : qui hait César?

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ¬ *pompeien*(*x1*) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. *loyal*(*x2*, *f1*(*x2*))
6. ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨
   ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)
9. ¬ *hait*(*x7*,*Cesar*) ∨ *Rep*(*x7*)

La 9ème clause est obtenue comme suit :

- la clause à prouver est : ∃ x *hait*(*x*,*Cesar*)
- sa négation est : ∀ *x* ¬ *hait*(*x*,*Cesar*)
- ce qui donne après standaradisation des variables : ¬ *hait*(*x7*,*Cesar*)
- on ajoute : *Rep*(*x7*)

Exemple 3
Répondre à la question : qui hait César?

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ¬ *pompeien*(*x1*) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. *loyal*(*x2*, *f1*(*x2*))
6. ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨ ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)
9. ¬ *hait*(*x7*,*Cesar*) ∨ *Rep*(*x7*)
10. *romain*(*Marcus*) — 2, 3, {*x1*=*Marcus*}
11. *loyal*(*Marcus*,*Cesar*) ∨ *hait*(*Marcus*,*Cesar*) — 6, 10, {*x4*=*Marcus*}
12. *loyal*(*Marcus*,*Cesar*) ∨ *Rep*(*Marcus*) — 9, 11 {*x7*=*Marcus*}
13. ¬ *personne*(*Marcus*) ∨ ¬ *dirigeant*(*Cesar*) ∨ ¬ *assassiner*(*Marcus*,*Cesar*) ∨ *Rep*(*Marcus*) — 7, 12, {*x5*=*Marcus*, *x6*=*Cesar*}
14. ¬ *personne*(*Marcus*) ∨ ¬ *dirigeant*(*Cesar*) ∨ *Rep*(*Marcus*) — 8,13
15. ¬ *personne*(*Marcus*) ∨ *Rep*(*Marcus*) — 4,14
16. *Rep*(*Marcus*) — 1,15

Réponse : Marcus