Processus d'inférence

- Les processus d'inférence sont des processus qui permettent de déduire des formules qui sont des conséquences logiques d'autres formules.
- Un bon processus d'inférence doit être **correct** (sound)
  - Toute formule déduite d'un ensemble de formules doit être une conséquence logique de ces formules
- Un processus d'inférence doit idéalement être **complet**
  - Il est capable de déduire toute formule qui est une conséquence logique d'autres formules

Règles d'inférence

- Modus Ponens
  - A partir de f_1 et $f_1 \rightarrow f_2$, on déduit f_2
  - Exemple : si on a $(wampusAhead \wedge wampusAlive) \rightarrow shoot$ et on a $(wampusAhead \wedge wampusAlive)$ alors $shoot$ peut être inféré
- Instanciation universelle
  - A partir de $\forall x\ f_1$ on déduit f_2 obtenu de f_1 en remplaçant toutes les occurrences libres de x par un terme n'ayant pas de variable en commun avec f_1
  - Exemple : Tous les chiens sont des mammifères, Fido est un chien, donc Fido est un mammifère

Preuve par résolution

- Procédure générale pour faire de l'inférence
  - Modus Ponens et Instanciation universelle sont des cas particulier
- Cette procédure est correcte et complète (sous certaines conditions)

- On aura besoin des outils suivants :
  - Substitution
  - Unification
  - Transformation sous forme normale conjonctive

Substitution

- On définit un **littéral** comme un prédicat ou la négation d'un prédicat
  - Exemple : $p(x, y), \neg\, p(x, y)$
- On définit une **clause** comme un disjonction de littéraux
  - Exemple : $p(x, y) \mathrm{V} p(x, y, z) \mathrm{V} \neg p(x, z)$
- Une **substitution** est un ensemble (possiblement vide) de paires de la forme $x_i = ti$ où x_i est une variable et t_i est un terme et les x_i sont distincts
  - Exemple : $\{x = John, y = père(Mary)\}$

Substitution

- L'application d'une substitution $\theta = \{x_1 = t_1, \dots, x_n = t_n\}$ à un littéral α donne un littéral $\alpha\theta$ obtenu de α en remplaçant simultanément toute occurrence de x_i par t_i dans α, pour chaque paire $x_i = ti$
- $\alpha\theta$ est appelé **instance** de α pour θ
  - Exemple : $\alpha = p(x, y, f(a)), \qquad \theta = \{y = x, x = b\}$

$$\alpha\theta = p(b, x, f(a))$$

- Si C est la clause $\alpha_1 \text{V} \dots \text{V}\ \alpha_n$, $C\theta$ est la clause $\alpha_1\theta \text{V} \dots \text{V}\ \alpha_n\theta$

Composition de substitutions

- Quelle serait la substitution équivalent à l'application successive de deux substitution $\theta = \{ x_1 = s_1, \ldots, x_m = s_m \}$ et $\sigma = \{ y_1 = t_1, \ldots, y_n = t_n \}$
  - on note une telle **composition** $\theta\sigma$

- La composition $\theta\sigma$ de θ et σ est la substitution obtenue comme suit :
  1. construire l'ensemble
     $$\{ x_1 = s_1\sigma, \ldots, x_m = s_m\sigma,\ y_1 = t_1, \ldots, y_m = t_n \}$$
     en appliquant σ à tous les termes s_i
  2. supprimer toutes les paires $y_i = t_i$ telles que $y_i \in \{x_1, \ldots, x_m\}$
  3. supprimer les identités, c-à-d., les paires pour lesquelles $s_i\sigma$ est devenu x_i

Composition de substitutions

- Exemple :

- Composition de $\theta = \{ x = f(y), y = z \}$ et $\sigma = \{ x = A, y = B, z = y \}$

1. $\{ x = f(B), y = y, x = A, y = B, z = y \}$
2. $\{ x = f(B), y = y, \sout{x = A}, \sout{y = B}, z = y \}$
3. $\{ x = f(B), \sout{y = y}, \sout{x = A}, \sout{y = B}, z = y \}$

Résultat: $\theta\sigma = \{ x = f(B), z = y \}$

Propriétés des substitutions

- La substitution identitée, notée ε, est l'ensemble vide
- $\theta\varepsilon = \theta$, pour toute substitution θ
- $(\alpha\sigma)\theta = \alpha(\sigma\theta)$, pour toute clause α et substitutions θ et σ
- $(\theta\sigma)\gamma = \theta(\sigma\gamma)$, pour toutes substitutions θ, σ et γ

Unification

- Soit S = $\{\alpha_1, \alpha_2\}$ une paire de 2 littéraux, on aimerait trouver une substitution θ qui **unifie** α_1 et α_2, c.-à-d. telle que $\alpha_1\theta = \alpha_2\theta$

  - ex. : { $p(f(x),z)$, $p(y,A)$ } sont unifiés par $\theta = \{y = f(x), z = A\}$ (A est un constante)

    $p(f(x),z)\ \theta = \mathbf{p(f(x),A)}$ et $p(y,A)\ \theta = \mathbf{p(f(x),A)}$

  - ex. : { $p(f(x),A)$, $p(y,f(w))$ } ne sont pas unifiables, puisqu'on ne peut pas substituer la constante A par $f(w)$

Unificateur le plus général

- Un **unificateur** θ de *S* est appelé **unificateur le plus général** (**UPG**) de *S* si pour tout unificateur σ de *S*, il existe une substitution γ telle que σ = θγ
  - ex. : θ = {*y* = *f*(*x*), *z* = *A*} est un UPG pour { *p*(*f*(*x*),*z*), *p*(*y*,*A*) }
    - *p*(*f*(*x*),*z*) θ = *p*(*y*,*A*) θ = *p*(*f*(*x*),*A*)
  - ex. : σ = {*y* = *f*(*A*), *x* = *A*, *z* = *A*} est unificateur mais pas UPG pour { *p*(*f*(*x*),*z*), *p*(*y*,*A*) }
    - *p*(*f*(*x*),*z*) σ = *p*(*y*,*A*) σ = *p*(*f*(*A*),*A*)
  - la substitution γ = {*x* = *A*} permet d'obtenir σ = θ γ = {*y* = *f*(*A*), *x* = *A*, *z* = *A*}
  - aucune substitution γ permet d'obtenir θ = σ γ
- On appelle **ensemble de désaccord** entre deux littéraux la paire des premiers termes des deux littéraux qui diffèrent (à partir de la gauche)
  - { *p*(*f*(*x*),*z*), *p*(*y*,*A*) } : l'ensemble de désaccord est { *f*(*x*), *y* }
  - { *p*(*f*(*x*),*z*), *p*(*f*(*x*),*A*) } : l'ensemble de désaccord est { *z*, *A* }

Unificateur le plus général

Algorithme UNIFICATEUR(S)

1. k=1; $\sigma_1 = \varepsilon$
2. **Si** σ_k est unificateur pour S,
   Alors retourner σ_k comme UPG de S
   Sinon calculer D_k l'ensemble de désaccord de $S\sigma_k$
3. **Si** il existe une paire (v, t) telle que v est une **variable** dans D_k qui n'apparaît pas dans t et $\{v = t\}$ est un unificateur pour D_k,
   alors $\sigma_{k+1} = \sigma_k\{v = t\}$, k=k+1;
   retourner à 2.
   Sinon exit; S n'est pas unifiable.

Unificateur le plus général : Exemple 1

- Trouver l'UPG de $p(x, f(x), y)$ et $p(y, z, u)$

- **Itération k=1**
  1. $\sigma_1 = \varepsilon = \{\}$ (σ_k est la valeur courante de l'UPG que l'on construit)

  2. σ_1 unifie-t-elle $p(x, f(x), y)$ et $p(y, z, u)$
     - » **non** : $p(x, f(x), y)\ \sigma_1 \rightarrow p(x, f(x), y) \neq p(y, z, u) \leftarrow p(y, z, u)\ \sigma_1$
     - » alors cherche ensemble de désaccord $D_1 = \{x, y\}$

  3. Existe-t-il un substitution qui unifie les éléments de D_1
     - » **oui** : $\{x = y\}$ (on aurait aussi pu choisir $\{y = x\}$ à la place)
     - » alors met à jour UPG : $\sigma_2 = \sigma_1 \{x = y\} = \{x = y\}$

Unificateur le plus général : Exemple 1

- Trouver l'UPG de $p(x, f(x), y)$ et $p(y, z, u)$

  - **Itération k=2**
    1. $\sigma_2 = \{x = y\}$

    2. σ_2 unifie-t-elle $p(x, f(x), y)$ et $p(y, z, u)$
       - » **non** : $p(x, f(x), y)\ \sigma_2 \rightarrow p(y, f(y), y) \neq p(y, z, u) \leftarrow p(y, z, u)\ \sigma_2$
       - » alors cherche ensemble de désaccord $D_2 = \{f(y), z\}$

    3. Existe-t-il un substitution qui unifie les éléments de D_2
       - » **oui** : $\{z = f(y)\}$
       - » alors met à jour UPG : $\sigma_3 = \sigma_2\{z = f(y)\} = \{x = y, z = f(y)\}$

Unificateur le plus général : Exemple 1

- Trouver l'UPG de $p(x, f(x), y)$ et $p(y, z, u)$

  - **Itération k=3**
    1. $\sigma_3 = \{x = y, z = f(y)\}$

    2. σ_3 unifie-t-elle $p(x, f(x), y)$ et $p(y, z, u)$
       - » **non** : $p(x, f(x), y)\ \sigma_3 \rightarrow p(y, f(y), y) \neq p(y, f(y), u) \leftarrow p(y, z, u)\ \sigma_3$
       - » alors cherche ensemble de désaccord $D_3 = \{y, u\}$

    3. Existe-t-il un substitution qui unifie les éléments de D_3
       - » **oui** : $\{y = u\}$ (on aurait aussi pu choisir $\{u = y\}$ à la place)
       - » alors met à jour UPG : $\sigma_4 = \sigma_3\{y = u\} = \{x = u, z = f(u), y = u\}$

Unificateur le plus général : Exemple 1

- Trouver l'UPG de $p(x, f(x), y)$ et $p(y, z, u)$

  - **Itération k=4**

    1. $\sigma_4 = \{x = u, z = f(u), y = u\}$

    2. σ_4 unifie-t-elle $p(x, f(x), y)$ et $p(y, z, u)$
       - » **oui** : $p(x, f(x), y)\ \sigma_4 \rightarrow p(u, f(u), u) = p(u, f(u), u) \leftarrow p(y, z, u)\ \sigma_4$
       - » alors on retourne l'UPG σ_4

Unificateur le plus général : Exemple 2

- Trouver l'UPG de $p(f(y), x)$ et $p(x, y)$

?

Mettre une formule sous forme normale conjonctive

(9 étapes)

1. **Élimination de l'implication**
   - Utiliser l'équivalence
     $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$
     pour enlever toutes les implications de la formule

2. **Réduire la portée de ¬**
   - Utiliser les **lois de Morgan**, c-à-d.
     i. $\neg (f_1 \vee f_2) \equiv \neg f_1 \wedge \neg f_2$
     ii. $\neg (f_1 \wedge f_2) \equiv \neg f_1 \vee \neg f_2$
     iii. $\neg \neg f \equiv f$,
     de sorte que ¬ est toujours suivi d'un prédicat

3. **Standardiser les variables**
   - renommer les variables de telle sorte qu'aucune paire de quantificateurs ne porte sur la même variable

Mettre une formule sous forme normale conjonctive

4. **Éliminer les quantificateurs existentiels**
   - chaque quantificateur existentiel est éliminé, en **remplaçant sa variable par une fonction des quantificateurs universels englobants**
     - ex. : $\forall x \forall y \exists z\, p(x, y, z)$ est remplacé par $\forall x \forall y\, p(x, y, f(x,y))$
     - on appelle ces fonctions (ex. $f(x,y)$ ci-haut) des **fonctions de Skolem**
     - le symbole de la fonction doit être unique (ne pas utiliser f à chaque fois)
     - si aucun argument, on utilise une constante unique
       - ex. : $\exists x\, q(x)$ devient $q(a)$ (où a n'est pas une constante déjà définie)

5. **Mettre en forme prénexe**
   - mettre tous les quantificateurs universels en tête

6. **Distribuer les disjonctions dans les conjonctions**
   - mettre sous forme de conjonction ($\wedge$) de disjonctions ($\vee$) de littéraux, en utilisant les équivalences de distributivité :

$$f_1 \vee (f_2 \wedge f_3) \equiv (f_1 \vee f_2) \wedge (f_1 \vee f_3)$$

Mettre une formule sous forme normale conjonctive

7. **Éliminer les symboles de quantificateurs universels**
   - on ne laisse que les variables

8. **Éliminer les conjonctions (∧)**
   - on génère des clauses séparées (sur des lignes différentes)

9. **Standardiser les variables à part**
   - renommer les variables de telle sorte que deux clauses différentes n'aient pas les mêmes variables

F.N.C

Exemple

1. Marcus est une personne.	1. $personne(Marcus)$
2. Marcus est un pompéien.	2. $pompeien(Marcus)$
3. Tous les pompéiens sont des romains.	3. $\forall x\ pompeien(x) \rightarrow romain(x)$
4. César est un dirigeant.	4. $dirigeant(Cesar)$
5. Tout le monde est loyal à quelqu'un.	5. $\forall x \exists y\ loyal(x,y)$
6. Tous les romains sont loyaux à César ou le haïssent.	6. $\forall x\ romain(x) \rightarrow loyal(x,Cesar) \vee hait(x,Cesar)$
7. Les seuls dirigeants qu'une personne essaie d'assassiner sont ceux auxquels elle n'est pas loyal	7. $\forall x \forall y\ personne(x) \wedge dirigeant(y) \wedge assassiner(x,y) \rightarrow \neg\ loyal(x,y)$
8. Marcus a essayer d'assassiner César.	8. $assassiner(Marcus,Cesar)$

Etape 1 : éliminer l'implication

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*)

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

Etape 2 : réduire la portée de ¬

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

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

Etape 3 : standardiser les variables

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

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. $\forall$ *x1* $\neg$ *pompeien*(*x1*) $\vee$ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. $\forall$ *x2* $\exists$ *x3* *loyal*(*x2*,*x3*)
6. $\forall$ *x4* $\neg$ *romain*(*x4*) $\vee$ *loyal*(*x4*,*Cesar*) $\vee$ *hait*(*x4*,*Cesar*)
7. $\forall$ *x5* $\forall$ *x6* $\neg$ *personne*(*x5*) $\vee$ $\neg$ *dirigeant*(*x6*) $\vee$ $\neg$ *assassiner*(*x5*,*x6*) $\vee$ $\neg$ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)

Etape 4 : éliminer les quantificateurs existentiels

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. $\forall$ *x1* $\neg$ *pompeien*(x1) $\vee$ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. $\forall$ *x2* $\exists$ *x3 loyal*(*x2*,*x3*)
6. $\forall$ *x4* $\neg$ *romain*(*x4*) $\vee$ *loyal*(*x4*,*Cesar*) $\vee$ *hait*(*x4*,*Cesar*)
7. $\forall$ *x5* $\forall$ *x6* $\neg$ *personne*(*x5*) $\vee$ $\neg$ *dirigeant*(*x6*) $\vee$ $\neg$ *assassiner*(*x5*,*x6*) $\vee$ $\neg$ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)

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

Etape 5 : mettre les formules en forme prénexe

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ∀ *x1* ¬ *pompeien*(x1) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. ∀ *x2* ∃ *x3* *loyal*(*x2*,*x3*)
6. ∀ *x4* ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ∀ *x5* ∀ *x6* ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨ ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)

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

Pas de changement ici

Etape 6 : distribuer les disjonctions dans les conjonctions

1. *personne*(*Marcus*)
2. *pompeien*(*Marcus*)
3. ∀ *x1* ¬ *pompeien*(x1) ∨ *romain*(*x1*)
4. *dirigeant*(*Cesar*)
5. ∀ *x2* ∃ *x3* *loyal*(*x2*,*x3*)
6. ∀ *x4* ¬ *romain*(*x4*) ∨ *loyal*(*x4*,*Cesar*) ∨ *hait*(*x4*,*Cesar*)
7. ∀ *x5* ∀ *x6* ¬ *personne*(*x5*) ∨ ¬ *dirigeant*(*x6*) ∨ ¬ *assassiner*(*x5*,*x6*) ∨ ¬ *loyal*(*x5*,*x6*)
8. *assassiner*(*Marcus*,*Cesar*)

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

Pas de changement ici

Etape 7 : éliminer les quantificateurs universels

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

Etape 8 : éliminer les conjonctions

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

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

Pas de changement ici

Etape 9 : standardiser les variables

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

Forme normale conjonctive

Pas de changement ici

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 certaines conditions)
- On aura besoin des outils suivants :
  - la **substitution**
  - l'**unification**
  - la **transformation sous forme normale conjonctive**

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 $\neg f_1 \vee f_2$

Preuve par résolution

Exemple

Base des faits (langue naturelle)

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

Forme logique (ordre 1)

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 : $hait(Marcus,Cesar)$

Forme normale conjonctive

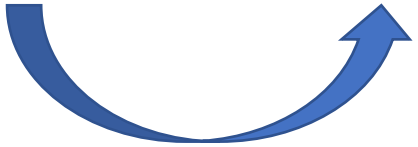
1. *personne(Marcus)*
2. *pompeien(Marcus)*
3. $\forall$ x *pompeien(x)* $\rightarrow$ *romain(x)*
4. *dirigeant(Cesar)*
5. $\forall$ *x* $\exists$ *y loyal(x,y)*
6. $\forall$ *x romain(x)* $\rightarrow$ *loyal(x,Cesar)* $\vee$ *hait(x,Cesar)*
7. $\forall$ x $\forall$ y *personne(x)* $\wedge$ *dirigeant(y)*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

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

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)*



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

Nous avons pu prouver que Marcus hait Cesar : hait(Marcus, Cesar)

Exercice : soit l'énoncé suivant

- Tous les chiens sont des animaux
- Tous les animaux vont mourir
- Fido est un chien

Utiliser la preuve par résolution pour prouver que « Fido va mourir »