

Déduction automatique en logique du premier ordre classique

David Delahaye

Faculté des Sciences
David.Delahaye@lirmm.fr

Master Informatique M2 2021-2022

Définition

- Une substitution σ est une application de \mathbb{V} vers \mathcal{T} telle que l'ensemble $dom(\sigma) = \{x \in \mathbb{V} \mid \sigma(x) \neq x\}$, appelé le domaine de σ , est fini ;
- L'image de σ est par définition $ran(\sigma) = \{\sigma(x) \mid x \in dom(\sigma)\}$ et on pose $yield(\sigma) = \bigcup_{t \in ran(\sigma)} FV(t)$;
- Nous écrivons aussi σ sous la forme $[\sigma(x_1)/x_1, \dots, \sigma(x_n)/x_n]$, où x_1, \dots, x_n contiennent toutes les variables de $dom(\sigma)$ et sont distinctes deux à deux ;
- En particulier, $[]$ est la substitution vide (ou identité).

Substitution sur les termes

Définition

- La notion de substitution s'étend aux termes et se note $t\sigma$ (de manière postfixe), où t est un terme et σ une substitution ;
- Elle se définit par récurrence structurelle sur les termes :
 - ▶ Si $x \in \mathbb{V}$ alors $x\sigma = \sigma(x)$;
 - ▶ Si $f \in \mathcal{F}$ et $t_1, \dots, t_n \in \mathcal{T}$ alors $f(t_1, \dots, t_n)\sigma = f(t_1\sigma, \dots, t_n\sigma)$.

Exemples

- $f(x, g(y, z))[a/x, h(b)/y, c/z] = f(a, g(h(b), c))$;
- $f(x, g(y, z))[a/x, h(b)/y] = f(a, g(h(b), z))$.

Substitution sur les termes

Définition

- La notion de substitution s'étend aux termes et se note $t\sigma$ (de manière postfixe), où t est un terme et σ une substitution ;
- Elle se définit par récurrence structurelle sur les termes :
 - ▶ Si $x \in \mathbb{V}$ alors $x\sigma = \sigma(x)$;
 - ▶ Si $f \in \mathcal{F}$ et $t_1, \dots, t_n \in \mathcal{T}$ alors $f(t_1, \dots, t_n)\sigma = f(t_1\sigma, \dots, t_n\sigma)$.

Exemples

- $f(x, g(y, z))[a/x, h(b)/y, c/z] = f(a, g(h(b), c))$;
- $f(x, g(y, z))[a/x, h(b)/y] = f(a, g(h(b), z))$.

Substitution sur les termes

Instances, composition, généralité

- Les instances d'un terme t sont tous les termes de la forme $t\sigma$, où σ est une substitution ;
- La composition $\sigma\sigma'$ de deux substitutions est définie par $t(\sigma\sigma') = (t\sigma)\sigma'$, où t est un terme ;
- Une substitution σ' est dite moins générale que σ , ce que nous notons $\sigma' \leq \sigma$, si et seulement s'il existe une substitution σ'' telle que $\sigma\sigma'' = \sigma'$.

Substitution sur les formules

Définition

- La notion de substitution s'étend aux formules et se note $\Phi\sigma$ (de manière postfixe), où Φ est un terme et σ une substitution ;
- Elle se définit par récurrence structurelle sur les formules :
 - ▶ Si $P \in \mathcal{S}_p$ d'arité n et $t_1, \dots, t_n \in \mathcal{T}$ alors
 $P(t_1, \dots, t_n)\sigma = P(t_1\sigma, \dots, t_n\sigma)$;
 - ▶ $\perp\sigma = \perp$, $\top\sigma = \top$;
 - ▶ Si $\Phi \in \mathcal{F}$ alors $(\neg\Phi)\sigma = \neg(\Phi\sigma)$;
 - ▶ Si $\Phi, \Phi' \in \mathcal{F}$ alors $(\Phi \oplus \Phi')\sigma = \Phi\sigma \oplus \Phi'\sigma$, où $\oplus \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$;
 - ▶ Si $x \in \mathcal{V}$ et $\Phi \in \mathcal{F}$ alors $(Qx.\Phi)\sigma = Qx'.\Phi[x'/x]\sigma$, où $Q \in \{\forall, \exists\}$ et $x' \notin \text{dom}(\sigma) \cup \text{yield}(\sigma) \cup (FV(\Phi) \setminus \{x\})$.

Exemples

- $P(f(x), g(y, z))[a/x, h(b)/y, c/z] = P(f(a), g(h(b), c))$;
- $(\forall x.P(f(x), g(y, z)))[a/x, h(b)/y, c/z] = \forall x'.P(f(x'), g(h(b), c))$.

Substitution sur les formules

Définition

- La notion de substitution s'étend aux formules et se note $\Phi\sigma$ (de manière postfixe), où Φ est un terme et σ une substitution ;
- Elle se définit par récurrence structurelle sur les formules :
 - ▶ Si $P \in \mathcal{S}_p$ d'arité n et $t_1, \dots, t_n \in \mathcal{T}$ alors
 $P(t_1, \dots, t_n)\sigma = P(t_1\sigma, \dots, t_n\sigma)$;
 - ▶ $\perp\sigma = \perp$, $\top\sigma = \top$;
 - ▶ Si $\Phi \in \mathcal{F}$ alors $(\neg\Phi)\sigma = \neg(\Phi\sigma)$;
 - ▶ Si $\Phi, \Phi' \in \mathcal{F}$ alors $(\Phi \oplus \Phi')\sigma = \Phi\sigma \oplus \Phi'\sigma$, où $\oplus \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$;
 - ▶ Si $x \in \mathcal{V}$ et $\Phi \in \mathcal{F}$ alors $(Qx.\Phi)\sigma = Qx'.\Phi[x'/x]\sigma$, où $Q \in \{\forall, \exists\}$ et $x' \notin \text{dom}(\sigma) \cup \text{yield}(\sigma) \cup (FV(\Phi) \setminus \{x\})$.

Exemples

- $P(f(x), g(y, z))[a/x, h(b)/y, c/z] = P(f(a), g(h(b), c))$;
- $(\forall x.P(f(x), g(y, z)))[a/x, h(b)/y, c/z] = \forall x'.P(f(x'), g(h(b), c))$.

Unification

Définition

- Une substitution σ est un unificateur de deux termes s et t si et seulement si $s\sigma = t\sigma$;
- Une substitution ρ est un renommage si et seulement si elle envoie des variables vers des variables, et elle est bijective.

Exemples

- $f(x, g(a))$ et $f(b, y)$ sont unifiables, l'unificateur est $[b/x, g(a)/y]$;
- $\rho = [x'/x, y'/y]$ est un renommage : $f(x, g(y))\rho = f(x', g(y'))$.

Unification

Définition

- Une substitution σ est un unificateur de deux termes s et t si et seulement si $s\sigma = t\sigma$;
- Une substitution ρ est un renommage si et seulement si elle envoie des variables vers des variables, et elle est bijective.

Exemples

- $f(x, g(a))$ et $f(b, y)$ sont unifiables, l'unificateur est $[b/x, g(a)/y]$;
- $\rho = [x'/x, y'/y]$ est un renommage : $f(x, g(y))\rho = f(x', g(y'))$.

Unification

Théorème

- Étant donnés deux termes s et t , soit il n'existe aucun unificateur de s et t , et nous disons alors que s et t ne sont pas unifiables, ou il existe un unificateur le plus général ou *mgu* ("most general unifier" en anglais), c'est-à-dire une substitution σ telle que :
 - ▶ σ est un unificateur de s et t , autrement dit, $s\sigma = t\sigma$;
 - ▶ Tout unificateur σ' de s et t est une instance de σ , c'est-à-dire qu'il existe une substitution σ'' telle que $\sigma' = \sigma\sigma''$.

Question

- Comment calculer le *mgu* de deux termes ?

Unification

Théorème

- Étant donnés deux termes s et t , soit il n'existe aucun unificateur de s et t , et nous disons alors que s et t ne sont pas unifiables, ou il existe un unificateur le plus général ou *mgu* ("most general unifier" en anglais), c'est-à-dire une substitution σ telle que :
 - ▶ σ est un unificateur de s et t , autrement dit, $s\sigma = t\sigma$;
 - ▶ Tout unificateur σ' de s et t est une instance de σ , c'est-à-dire qu'il existe une substitution σ'' telle que $\sigma' = \sigma\sigma''$.

Question

- Comment calculer le *mgu* de deux termes ?

Algorithme d'unification de Robinson

- $G\{s_1 = t_1, \dots, s_n = t_n\} \hookrightarrow \{x_1 = u_1, \dots, x_m = u_m\}$
où x_i sont des variables distinctes et $x_i \notin u_j$;
- Règles :
 - ▶ $G \cup \{t = t\} \hookrightarrow G$ (delete);
 - ▶ $G \cup \{f(s_1, \dots, s_n) = f(t_1, \dots, t_n)\} \hookrightarrow G \cup \{s_1 = t_1, \dots, s_n = t_n\}$ (decompose);
 - ▶ $G \cup \{f(s_1, \dots, s_n) = g(t_1, \dots, t_m)\} \hookrightarrow \perp$, si $f \neq g$ ou $n \neq m$ (conflict);
 - ▶ $G \cup \{f(s_1, \dots, s_n) = x\} \hookrightarrow G \cup \{x = f(s_1, \dots, s_n)\}$ (swap);
 - ▶ $G \cup \{x = t\} \hookrightarrow G[t/x] \cup \{x = t\}$, si $x \notin t$ et $x \in G$ (eliminate);
 - ▶ $G \cup \{x = f(s_1, \dots, s_n)\} \hookrightarrow \perp$, si $x \in f(s_1, \dots, s_n)$ (check).

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, g(a)) = f(b, y)\}$;
- $\{f(x, g(a)) = f(b, y)\} \xrightarrow{\text{decompose}} \{x = b, g(a) = y\} \xrightarrow{\text{swap}} \{x = b, y = g(a)\}$;
- $\text{mgu}(f(x, g(a)), f(b, y)) = [b/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, g(a)) = f(b, y)\}$;
- $\{f(x, g(a)) = f(b, y)\} \hookrightarrow_{\text{decompose}}$
 $\{x = b, g(a) = y\} \hookrightarrow_{\text{swap}}$
 $\{x = b, y = g(a)\}$;
- $\text{mgu}(f(x, g(a)), f(b, y)) = [b/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, g(a)) = f(b, y)\}$;
- $\{f(x, g(a)) = f(b, y)\} \hookrightarrow_{\text{decompose}} \{x = b, g(a) = y\} \hookrightarrow_{\text{swap}} \{x = b, y = g(a)\}$;
- $\text{mgu}(f(x, g(a)), f(b, y)) = [b/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, g(a)) = f(b, y)\}$;
- $\{f(x, g(a)) = f(b, y)\} \hookrightarrow_{\text{decompose}} \{x = b, g(a) = y\} \hookrightarrow_{\text{swap}} \{x = b, y = g(a)\}$;
- $\text{mgu}(f(x, g(a)), f(b, y)) = [b/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, g(a)) = f(b, y)\}$;
- $\{f(x, g(a)) = f(b, y)\} \hookrightarrow_{\text{decompose}} \{x = b, g(a) = y\} \hookrightarrow_{\text{swap}} \{x = b, y = g(a)\}$;
- $\text{mgu}(f(x, g(a)), f(b, y)) = [b/x, g(a)/y]$.

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, g(a)) = f(b, y)\}$;
- $\{f(x, g(a)) = f(b, y)\} \hookrightarrow_{\text{decompose}} \{x = b, g(a) = y\} \hookrightarrow_{\text{swap}} \{x = b, y = g(a)\}$;
- $\text{mgu}(f(x, g(a)), f(b, y)) = [b/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $mgu(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $\text{mgu}(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $mgu(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $mgu(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $mgu(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $mgu(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $\text{mgu}(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(g(x), x) = f(y, a)\}$;
- $\{f(g(x), x) = f(y, a)\} \hookrightarrow_{\text{decompose}}$
 $\{g(x) = y, x = a\} \hookrightarrow_{\text{swap}}$
 $\{y = g(x), x = a\} \hookrightarrow_{\text{eliminate}}$
 $\{y = g(a), x = a\}$;
- $\text{mgu}(f(g(x), x), f(y, a)) = [a/x, g(a)/y]$.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Unification

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Algorithme d'unification de Robinson

- Ensemble initial d'équations : $\{f(x, b) = f(a, x)\}$;
- $\{f(x, b) = f(a, x)\} \hookrightarrow_{\text{decompose}}$
 $\{x = a, b = x\} \hookrightarrow_{\text{eliminate}}$
 $\{x = a, b = a\} \hookrightarrow_{\text{conflict}}$
 \perp ;
- $f(x, b)$ et $f(a, x)$ ne sont pas unifiables.

Extension aux atomes des formules

- Deux atomes $P(t_1, \dots, t_n)$ et $Q(t'_1, \dots, t'_m)$, avec $P, Q \in \mathcal{P}$ et $t_1, \dots, t_n, t'_1, \dots, t'_m \in \mathcal{T}$ sont unifiables si et seulement si :
 - ▶ $P = Q$ (même symbole de prédicat) ;
 - ▶ $n = m$ (même arité) ;
 - ▶ $\{t_1 = t'_1, \dots, t_n = t'_n\} \not\vdash \perp$ (on a une solution).

Méthode des tableaux

Un peu d'histoire

- Méthode plus ancienne que la résolution ;
- Introduite par les pionniers Hintikka et Beth (années 50) ;
- Perfectionnée ensuite par Smullyan et Fitting ;
- À partir du calcul des séquents de Gentzen sans coupure.

Principe

- Par réfutation sur la proposition initiale et par cas ;
- Sans ou avec variables libres (métavariabes) ;
- Avec skolémisation ou ϵ -termes.
- En version destructive ou non destructive.

Méthode des tableaux

Règles de clôture et règles analytiques

$$\frac{\perp}{\odot} \odot \perp$$

$$\frac{\neg \top}{\odot} \odot \neg \top$$

$$\frac{P \quad \neg P}{\odot} \odot$$

$$\frac{\neg \neg P}{P} \alpha_{\neg \neg}$$

$$\frac{P \Leftrightarrow Q}{\neg P, \neg Q \mid P, Q} \beta_{\Leftrightarrow}$$

$$\frac{\neg(P \Leftrightarrow Q)}{\neg P, Q \mid P, \neg Q} \beta_{\neg \Leftrightarrow}$$

$$\frac{P \wedge Q}{P, Q} \alpha_{\wedge}$$

$$\frac{\neg(P \vee Q)}{\neg P, \neg Q} \alpha_{\neg \vee}$$

$$\frac{\neg(P \Rightarrow Q)}{P, \neg Q} \alpha_{\neg \Rightarrow}$$

$$\frac{P \vee Q}{P \mid Q} \beta_{\vee}$$

$$\frac{\neg(P \wedge Q)}{\neg P \mid \neg Q} \beta_{\neg \wedge}$$

$$\frac{P \Rightarrow Q}{\neg P \mid Q} \beta_{\Rightarrow}$$

Méthode des tableaux (sans variable libre)

δ/γ -règles

$$\frac{\exists x.P(x)}{P(c)} \delta_{\exists}, \text{ c frais}$$

$$\frac{\neg \forall x.P(x)}{\neg P(c)} \delta_{\neg \forall}, \text{ c frais}$$

$$\frac{\forall x.P(x)}{P(t)} \gamma_{\forall \text{inst}}$$

$$\frac{\neg \exists x.P(x)}{\neg P(t)} \gamma_{\neg \exists \text{inst}}$$

Méthode des tableaux (avec variable libre, destructif)

δ/γ -règles

$$\frac{\exists x.P(x)}{P(f(X_1, \dots, X_n))} \delta_{\exists}, \begin{array}{l} f \text{ frais,} \\ X_i \text{ var. lib.} \end{array}$$

$$\frac{\neg \forall x.P(x)}{\neg P(f(X_1, \dots, X_n))} \delta_{\neg \forall}, \begin{array}{l} f \text{ frais,} \\ X_i \text{ var. lib.} \end{array}$$

$$\frac{\forall x.P(x)}{P(X)} \gamma_{\forall M}$$

$$\frac{\neg \exists x.P(x)}{\neg P(X)} \gamma_{\neg \exists M}$$

$$\frac{\forall x.P(x)}{P(t)} \gamma_{\forall \text{inst}}$$

$$\frac{\neg \exists x.P(x)}{\neg P(t)} \gamma_{\neg \exists \text{inst}}$$

Appliquer σ à l'arbre s'il existe dans la branche
deux littéraux K et $\neg L$ t.q. $\sigma = mgu(K, L)$

$$\frac{}{\odot} \odot$$

Méthode des tableaux (avec variable libre, non destructif)

δ/γ -règles

$$\frac{\exists x.P(x)}{P(f(X_1, \dots, X_n))} \delta_{\exists}, \quad \begin{array}{l} f \text{ frais,} \\ X_i \text{ var. lib.} \end{array} \quad \frac{\neg \forall x.P(x)}{\neg P(f(X_1, \dots, X_n))} \delta_{\neg \forall}, \quad \begin{array}{l} f \text{ frais,} \\ X_i \text{ var. lib.} \end{array}$$

$$\frac{\forall x.P(x)}{P(X)} \gamma_{\forall M}$$

$$\frac{\neg \exists x.P(x)}{\neg P(X)} \gamma_{\neg \exists M}$$

$$\frac{\forall x.P(x)}{P(t)} \gamma_{\forall \text{inst}}$$

$$\frac{\neg \exists x.P(x)}{\neg P(t)} \gamma_{\neg \exists \text{inst}}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

δ/γ -règles

$$\frac{\exists x.P(x)}{P(\epsilon(x).P(x))} \delta_{\exists}$$

$$\frac{\neg\forall x.P(x)}{\neg P(\epsilon(x).\neg P(x))} \delta_{\neg\forall}$$

$$\frac{\forall x.P(x)}{P(X)} \gamma_{\forall M}$$

$$\frac{\neg\exists x.P(x)}{\neg P(X)} \gamma_{\neg\exists M}$$

$$\frac{\forall x.P(x)}{P(t)} \gamma_{\forall inst}$$

$$\frac{\neg\exists x.P(x)}{\neg P(t)} \gamma_{\neg\exists inst}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

- Preuve de : $(\forall x. P(x) \vee Q(x)) \Rightarrow P(a) \vee Q(a)$;
- Réfutation : $\neg((\forall x. P(x) \vee Q(x)) \Rightarrow P(a) \vee Q(a))$;
- Premières règles ($\alpha_{\neg\Rightarrow}$, $\alpha_{\neg\vee}$) : $\forall x. P(x) \vee Q(x)$, $\neg P(a)$, $\neg Q(a)$.

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\underline{\forall x (P(x) \vee Q(x)) , \neg P(a) , \neg Q(a)}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\forall x (P(x) \vee Q(x)) , \neg P(a) , \neg Q(a)}{P(X) \vee Q(X)} \gamma_{\forall M}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\forall x (P(x) \vee Q(x)) , \neg P(a) , \neg Q(a)}{P(X) \vee Q(X)} \gamma_{\forall M}}{\frac{P(X)}{Q(X)}} \beta_{\vee}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\forall x (P(x) \vee Q(x)) , \neg P(a) , \neg Q(a)}{P(X) \vee Q(X)} \gamma_{\forall M}}{\frac{P(X)}{Q(X)} \beta_{\vee}}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\forall x (P(x) \vee Q(x)) , \neg P(a) , \neg Q(a)}{P(X) \vee Q(X)} \gamma_{\forall M}}{\frac{P(X)}{P(a) \vee Q(a)} \gamma_{\forall \text{inst}} \quad Q(X)} \beta_{\vee}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\frac{\forall x (P(x) \vee Q(x)) , \neg P(a) , \neg Q(a)}{P(X) \vee Q(X)} \gamma_{\forall M}}{P(X)} \gamma_{\forall \text{inst}}}{P(a) \vee Q(a)} \beta_{\vee}}{P(a) \quad Q(a)} \beta_{\vee}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\frac{\frac{P(a)}{\odot} \quad \odot}{Q(a)} \beta_V}{P(a) \vee Q(a)} \gamma_{\forall \text{inst}}}{P(X)} \quad \frac{Q(X)}{\beta_V}}{P(X) \vee Q(X)} \gamma_{\forall M} \quad \frac{\forall x (P(x) \vee Q(x)), \neg P(a), \neg Q(a)}{\quad} \gamma_{\forall M}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\frac{P(a)}{\odot} \quad \odot}{P(a) \vee Q(a)} \quad \odot \quad \frac{\frac{Q(a)}{\odot} \quad \odot}{Q(a)} \quad \odot}{P(a) \vee Q(a)} \quad \beta_V}{\frac{P(X)}{P(X)} \quad \gamma_{Vinst} \quad \frac{Q(X)}{Q(X)} \quad \beta_V} \quad \gamma_{\forall M}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\frac{P(a)}{\odot} \quad \odot}{P(a)} \quad \frac{\frac{Q(a)}{\odot} \quad \odot}{Q(a)}}{P(a) \vee Q(a)} \beta_V \quad \frac{P(X)}{\gamma_{\text{Vinst}}} \quad \frac{Q(X)}{\beta_V}}{P(X) \vee Q(X)} \gamma_{\forall M} \quad \frac{\forall x (P(x) \vee Q(x)), \neg P(a), \neg Q(a)}{\quad} \gamma_{\forall M}$$

Méthode des tableaux (non destructif, avec ϵ -termes)

Exemple

$$\frac{\frac{\frac{\frac{P(a)}{\odot}}{\odot} \quad \frac{Q(a)}{\odot}}{\odot} \beta_{\vee}}{P(a) \vee Q(a)} \quad \frac{\forall x (P(x) \vee Q(x)), \neg P(a), \neg Q(a)}{P(a) \vee Q(a)} \gamma_{\forall \text{inst}}$$

Résolution

Principe

- Par réfutation ;
- Nécessité de clausifier ;
- Obtention d'une formule universelle (skolémisation) ;
- Variables universelles \equiv métavariabes (unification).

Skolémisation

Formule existentielle/universelle

- Formule existentielle : $\exists x_1 \dots \exists x_n. P(x_1, \dots, x_n)$;
- Formule universelle : $\forall x_1 \dots \forall x_n. P(x_1, \dots, x_n)$.

Théorème de Herbrand-Skolem

Pour toute formule Φ :

- Il existe une formule existentielle Φ' t.q. Φ' est valide ssi Φ est valide (Φ' est une forme de Herbrand de Φ) ;
- Il existe une formule universelle Φ' t.q. Φ' est insatisfiable ssi Φ est insatisfiable (Φ' est une forme de Skolem de Φ).

Fonctions de skolémisation et herbrandisation

- Si Φ est atomique, $s(\Phi) = h(\Phi) = \Phi$;
- $s(\Phi \wedge \Phi') = s(\Phi) \wedge s(\Phi')$, $h(\Phi \wedge \Phi') = h(\Phi) \wedge h(\Phi')$;
- $s(\Phi \vee \Phi') = s(\Phi) \vee s(\Phi')$, $h(\Phi \vee \Phi') = h(\Phi) \vee h(\Phi')$;
- $s(\neg\Phi) = \neg h(\Phi)$, $h(\neg\Phi) = \neg s(\Phi)$;
- $s(\Phi \Rightarrow \Phi') = h(\Phi) \Rightarrow s(\Phi')$, $h(\Phi \Rightarrow \Phi') = s(\Phi) \Rightarrow h(\Phi')$;
- $s(\forall x.\Phi) = s(\Phi)$, $h(\forall x.\Phi) = h(\Phi)[f(x_1, \dots, x_n)/x]$, où x_1, \dots, x_n sont les variables libres de $\forall x.\Phi$;
- $s(\exists x.\Phi) = s(\Phi)[f(x_1, \dots, x_n)/x]$, où x_1, \dots, x_n sont les variables libres de $\exists x.\Phi$, $h(\exists x.\Phi) = h(\Phi)$.
- Ensuite, une fois le calcul terminé :
 - ▶ Skolémisation : $\forall x_1 \dots \forall x_n. s(\Phi)$, où x_1, \dots, x_n sont les variables libres de $s(\Phi)$;
 - ▶ Herbrandisation : $\exists x_1 \dots \exists x_n. h(\Phi)$, où x_1, \dots, x_n sont les variables libres de $h(\Phi)$.

Exemple

- Skolémisation de $\forall x. \exists y. \forall z. P(x, y, z)$;
- $s(\forall x. \exists y. \forall z. P(x, y, z)) =$
 $s(\exists y. \forall z. P(x, y, z)) =$
 $s(\forall z. P(x, y, z))[f(x)/y] =$
 $s(P(x, y, z))[f(x)/y] =$
 $P(x, y, z)[f(x)/y] =$
 $P(x, f(x), z)$;
- On obtient donc : $\forall x. \forall z. P(x, f(x), z)$.

Clausification

Principe

- On skolemise : on obtient une formule universelle $\forall \vec{x}. \Phi$;
- On élimine les quantificateurs, puis on met Φ en cnf.

Exemple

- $$\begin{aligned} s(\neg((\forall x. P(x) \vee Q(x)) \Rightarrow P(a) \vee Q(a))) &= \\ \neg(h((\forall x. P(x) \vee Q(x)) \Rightarrow P(a) \vee Q(a))) &= \\ \neg(s(\forall x. P(x) \vee Q(x)) \Rightarrow h(P(a) \vee Q(a))) &= \\ \neg(P(x) \vee Q(x) \Rightarrow P(a) \vee Q(a)) ; \end{aligned}$$
- $$\begin{aligned} \neg(P(x) \vee Q(x) \Rightarrow P(a) \vee Q(a)) &= \\ \neg(\neg(P(x) \vee Q(x)) \vee P(a) \vee Q(a)) &= \\ \neg(\neg(P(x) \vee Q(x))) \wedge \neg P(a) \wedge \neg Q(a) &= \\ (P(x) \vee Q(x)) \wedge \neg P(a) \wedge \neg Q(a) ; \end{aligned}$$
- $S = \{P(x) \vee Q(x), \neg P(a), \neg Q(a)\}.$

Résolution et factorisations binaires

Résolution binaire

$$\frac{A \vee C \quad \neg B \vee D}{\sigma(C) \vee \sigma(D)} \text{ res}$$

où $\sigma(A) = \sigma(B)$.

Factorisations binaires

$$\frac{A \vee B \vee C}{\sigma(B) \vee \sigma(C)} \text{ fact}^+ \qquad \frac{\neg A \vee \neg B \vee C}{\neg \sigma(B) \vee \sigma(C)} \text{ fact}^-$$

où $\sigma(A) = \sigma(B)$.

Attention aux noms des variables !

Un problème

- Est-ce que l'ensemble $S = \{P(x, a), \neg P(b, x)\}$ est insatisfiable ?
- Oui car les clauses sont universellement quantifiées :
 - $\forall x. P(x, a)$
 - $\forall x. \neg P(b, x)$
- Mais aucune règle de résolution ne s'applique !

Solution : renommage

- Avant de faire une résolution entre deux clauses C_1 et C_2 , on renomme les variables de C_1 et C_2 de manière à ce que les deux clauses ne partagent plus aucune variable ;
- On appelle ça la standardisation des variables (terme qui vient de la communauté programmation logique).

Attention aux noms des variables !

Un problème

- Est-ce que l'ensemble $S = \{P(x, a), \neg P(b, x)\}$ est insatisfiable ?
- Oui car les clauses sont universellement quantifiées :
 - ▶ $\forall x. P(x, a)$
 - ▶ $\forall x. \neg P(b, x)$
- Mais aucune règle de résolution ne s'applique !

Solution : renommage

- Avant de faire une résolution entre deux clauses C_1 et C_2 , on renomme les variables de C_1 et C_2 de manière à ce que les deux clauses ne partagent plus aucune variable ;
- On appelle ça la standardisation des variables (terme qui vient de la communauté programmation logique).

Attention aux noms des variables !

Un problème

- Est-ce que l'ensemble $S = \{P(x, a), \neg P(b, x)\}$ est insatisfiable ?
- Oui car les clauses sont universellement quantifiées :
 - ▶ $\forall x. P(x, a)$
 - ▶ $\forall x. \neg P(b, x)$
- Mais aucune règle de résolution ne s'applique !

Solution : renommage

- Avant de faire une résolution entre deux clauses C_1 et C_2 , on renomme les variables de C_1 et C_2 de manière à ce que les deux clauses ne partagent plus aucune variable ;
- On appelle ça la standardisation des variables (terme qui vient de la communauté programmation logique).

Attention aux noms des variables !

Un problème

- Est-ce que l'ensemble $S = \{P(x, a), \neg P(b, x)\}$ est insatisfiable ?
- Oui car les clauses sont universellement quantifiées :
 - ▶ $\forall x. P(x, a)$
 - ▶ $\forall x. \neg P(b, x)$
- Mais aucune règle de résolution ne s'applique !

Solution : renommage

- Avant de faire une résolution entre deux clauses C_1 et C_2 , on renomme les variables de C_1 et C_2 de manière à ce que les deux clauses ne partagent plus aucune variable ;
- On appelle ça la standardisation des variables (terme qui vient de la communauté programmation logique).

Attention aux noms des variables !

Un problème

- Est-ce que l'ensemble $S = \{P(x, a), \neg P(b, x)\}$ est insatisfiable ?
- Oui car les clauses sont universellement quantifiées :
 - ▶ $\forall x. P(x, a)$
 - ▶ $\forall x. \neg P(b, x)$
- Mais aucune règle de résolution ne s'applique !

Solution : renommage

- Avant de faire une résolution entre deux clauses C_1 et C_2 , on renomme les variables de C_1 et C_2 de manière à ce que les deux clauses ne partagent plus aucune variable ;
- On appelle ça la standardisation des variables (terme qui vient de la communauté programmation logique).

Procédure de résolution (la même qu'en propositionnel !)

Algorithme (« Given-Clause Algorithm »)

```
Sat :=  $\emptyset$  ;  
tant que  $S \neq \emptyset$  faire  
  choisir  $C \in S$  ;  
   $S := S \setminus \{C\}$  ;  
  si  $C = \square$  alors retourner « insatisfiable » ;  
  si  $C$  est une tautologie alors ; (* passer à la clause suivante *)  
  sinon, si  $C \in Sat$  alors ; (* idem *)  
  sinon pour tout résolvant  $C_1$  entre  $C$   
  et une clause de  $Sat \cup \{C\}$  faire  
     $S := S \cup \{C_1\}$  ;  
   $Sat := Sat \cup \{C\}$  ;  
retourner « satisfiable ».
```


Exemple

- Preuve de : $(\forall x. P(x) \vee Q(x)) \Rightarrow P(a) \vee Q(a)$;
- Clausification de $\neg((\forall x. P(x) \vee Q(x)) \Rightarrow P(a) \vee Q(a))$:
 $S = \{P(x) \vee Q(x), \neg P(a), \neg Q(a)\}$;
- Résolution entre $P(x) \vee Q(x)$ et $\neg P(a)$, $\sigma = [a/x]$: $Q(a)$;
- Résolution entre $Q(a)$ et $\neg Q(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
 - Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
 - Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
 - Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Pourquoi des règles de factorisation ?

Exemple

- Ensemble de clauses : $S = \{P(x) \vee P(y), \neg P(a) \vee \neg P(z)\}$;
- S est bien insatisfiable ;
- Résolution entre $P(x) \vee P(y)$ et $\neg P(a) \vee \neg P(z)$: $P(y) \vee \neg P(z)$;
- Résolution entre $P(x) \vee P(y)$ et $P(y') \vee \neg P(z)$: $P(y) \vee P(y')$;
- Résolution entre $\neg P(a) \vee \neg P(z)$ et $P(y) \vee \neg P(z')$: $\neg P(z) \vee \neg P(z')$;
- On régénère les mêmes clauses indéfiniment ;
- Règle fact^+ sur $P(x) \vee P(y)$: $P(x)$;
- Règle fact^- sur $\neg P(a) \vee \neg P(z)$: $\neg P(a)$;
- Résolution entre $P(x)$ et $\neg P(a)$: \square .

Propriétés de la résolution

Correction et complétude

- La résolution est correcte et complète : un ensemble de clauses S est insatisfiable ssi on peut dériver la clause vide à partir de S ;
- La correction est triviale car la conclusion de chaque règle de résolution est une conséquence logique de ses prémisses ;
- La complétude peut être démontrée en utilisant la complétude de la résolution propositionnelle et le « lifting ».

Quelques mots sur la complétude

Règles et méthode de recherche de preuve

- Ne pas confondre la complétude des règles de résolution et la complétude de l'algorithme de recherche de preuve ;
- Les règles de résolution sont complètes ;
- En revanche, l'algorithme doit être plus précis sur le choix de la clause pour établir sa complétude ;
- La résolution avec sélection arbitraire des clauses est incomplète ;
- La résolution ordonnée (avec un ordre sur les clauses) est complète.