4.2 La négation en programmation logique

Dans cette section nous allons étudier plusieurs notions de négation dans la programmation logique.

4.2.1 L'hypothèse du monde clos

Une première approche pour définir la négation en programme logique est de dire, étant donné un programme logique P, qu'un littéral négatif clos $\neg R(\bar{t})$ est vrai ssi et seulement ssi $R(\bar{t}) \notin \mathcal{M}_P$, c'est-à-dire s'il n'est pas le cas que $P \models R(\bar{t})$. Remarquer que $P \not\models R(\bar{t})$ n'est pas équivalent à $P \models \neg R(\bar{t})$ – en fait, puisque la base de Herbrand est un modèle de tout programme logique il n'est jamais possible qu'un littéral négatif soit une conséquence logique d'un programme logique.

Dans le domaine de l'Intelligence Artificielle, cette définition est justifié par l'hypothèse que notre programme exprime toutes les faites vrais. Cette hypothèse est connu sous le nom *Hypothèse du Monde Clos*, ou en anglais *Closed World Assumption*.

Le problème avec cette définition est que, par conséquence du résultat de l'exercice 85, l'ensemble des atomes clos qui ne sont pas vrais dans $\mathcal{M}_P$ est en général pas récursivement énumérable. Il n'est donc pas possible de réaliser un système de déduction qui est complet et correct pour toutes les conséquences négatives dans le sens de l'hypothèse du monde clos.

4.2.2 Négation par échec

Motivé par le traitement de la négation en Prolog (section **??**), on peut essayer de donner une définition plus faible de la négation :

Definition 4.2.1 *Soit P un programme logique. L'*ensemble d'échec fini *(angl. :* finite failure set*) de P est l'ensemble des atomes clos A tel qu'il y a un LD-arbre fini d'échec à partir de $P \cup \{\neg A\}$.*

L'ensemble d'échec fini d'un programme logique est toujours récursivement énumérable : on peut simplement construire en parallèle tous les LD-arbres à partir de $P \cup \{\neg A\}$.

Remarquons que cette définition de vérité d'un littéral négatif est différent du comportement de Prolog puisque Prolog essaye de construire un LD-arbre avec une stratégie spécifique et de le construire en profondeur d'abord, tandis que définition 4.2.1 parle de l'existence d'un arbre fini d'échec selon une stratégie quelconque.

Cette interprétation de la négation en programmation logique est fortement liée au point fixe *le plus grand* de l'opérateur de conséquence immédiate, comme expliqué dans la suite.

Nous avons vu dans la section 4.1.2 que $(HS(\mathcal{F},\mathcal{P}),\subseteq)$ est un cpo pointé, et même un treillis complet. Par dualité, $(HS(\mathcal{F},\mathcal{P}),\supseteq)$ est également un cpo pointé. Nous avons également vu que l'opérateur de conséquence immédiate T_P associé à un programme logique P est monotone, et même continu dans le cpo $(HS(\mathcal{F},\mathcal{P}),\subseteq)$.

Théorème 4.2.1 *[Knaster et Tarski] Tout opérateur monotone sur un cpo pointé possède un plus petit point fixe.*

Par conséquence, T_P possède un point fixe le plus petit en $(HS(\mathcal{F},\mathcal{P}),\supseteq)$, c'est-à-dire un point fixe le plus grand en $(HS(\mathcal{F},\mathcal{P}),\subseteq)$. Dans la suite nous font toujours référence au cpo $(HS(\mathcal{F},\mathcal{P}),\subseteq)$ et parlons d'un point fixe le plus grand ou le plus petit, et de continuité vers le haut et continuité vers le bas.

On général, on définit

$$\begin{array}{rcl} T_P\downarrow 0 &=& HB(\mathcal{F},\mathcal{P}) \\ T_P\downarrow \alpha+1 &=& T_P(T_P\downarrow\alpha) \\ T_P\downarrow\alpha &=& \bigcap\limits_{\beta<\alpha} T_P\downarrow\beta \quad \text{si } \alpha \text{ est un ordinal limite} \end{array}$$

La preuve du théorème 4.2.1 montre qu'il y a un ordinal α tel que $T_P\downarrow\alpha$ est le point fixe le plus grand de T_P. Puisque T_P n'est pas continu vers le bas (voir l'exemple 4.2.3), cet ordinal α peut être plus grand que ω.

Exemple 4.2.1 Nous reprenons le programme de l'exemple 4.1.3. On obtient pour le calcul du point fixe le plus grand :

$$\begin{array}{rcl} T_P\downarrow 0 &=& \{P(s^i(0),s^j(0),s^k(0)) \mid i,j,k\geq 0\} \\ T_P\downarrow 1 &=& \{P(0,s^j(0),s^j(0)) \mid j\geq 0\}\cup\{P(s^i(0),s^j(0),s^k(0)) \mid j\geq 0, i,k\geq 1\} \\ T_P\downarrow n &=& \{P(s^i(0),s^j(0),s^{i+j}(0)) \mid j\geq 0, 0\leq i\leq n\} \\ && \cup\ \{P(s^i(0),s^j(0),s^k(0)) \mid j\geq 0, i,k\geq n\} \qquad (n\geq 1) \\ T_P\downarrow\omega &=& \{P(s^i(0),s^j(0),s^{j+i}(0)) \mid i,j\geq 0\} \end{array}$$

Donc, $T_P\downarrow\omega$ est identique avec le point fixe le plus petit.

Exemple 4.2.2 Soit le programme suivant :

$$\begin{array}{rcl} P(x) &\leftarrow& P(x) \\ R(a) &\leftarrow& \\ R(f(x)) &\leftarrow& R(x) \end{array}$$

Le point fixe le plus petit est

$$\{R(f^n(a)) \mid n\geq 0\}$$

donc en particulier le prédicat P est toujours faux dans le point fixe le plus petit. Lepoint fixe le plus grand est

$$\{P(f^n(a) \mid n\geq 0\}\cup\{R(f^n(a)) \mid n\geq 0\}$$

Le point fixe le plus grand est $T_P\downarrow 0$.

Exemple 4.2.3 Soit le programme logique suivant :

$$\begin{array}{rcl} Q(a) &\leftarrow& R(x) \\ R(f(x)) &\leftarrow& R(x) \end{array}$$

Le calcul du point fixe le plus grand donne

$$\begin{aligned}
T_P \downarrow 0 &= \{Q(f^i(a)) \mid i \geq 0\} \cup \{R(f^i(a)) \mid i \geq 0\} \\
T_P \downarrow 1 &= \{Q(a)\} \cup \{R(f^i(a)) \mid i \geq 1\} \\
T_P \downarrow n &= \{Q(a)\} \cup \{R(f^i(a)) \mid i \geq n\} \quad (n \geq 1) \\
T_P \downarrow \omega &= \{Q(a)\} \\
T_P \downarrow \omega + 1 &= \emptyset
\end{aligned}$$

Le point fixe le plus grand est donc atteint en $\omega + 1$ étapes. Par conséquence, T_P n'est pas continu vers le bas (sinon, le point fixe le plus grand serait atteint en ω étapes).

Remarquez que $Q(a)$ n'est pas dans l'ensemble d'échec fini du programme.

Nous disons qu'un LD-arbre est *équitable* si sur ses branches il n'y a aucun littéral qui n'est jamais sélectionné (il est entendu qu'un littéral conserve son « identité » quand une substitution est appliquée au but). Par exemple, l'arbre de la figure 4.4 n'est pas équitable. En particulier, un SLD-arbre via une stratégie équitable est un LD-arbre équitable.

Théorème 4.2.2 *[Apt, van Emden, Lassez, Maher] Soit P un programme logique et A un atome clos. Les énoncés suivants sont équivalents :*

1. *A est dans l'ensemble d'échec fini de P.*
2. *$A \notin T_P \downarrow \omega$*
3. *Tout LD-arbre équitable pour la requête $\leftarrow A$ est un arbre fini d'échec.*

La preuve de la direction (1) $\Rightarrow$ (2) utilise les mêmes techniques que les preuves vues dans les sections 4.1.3 and 4.1.4. La preuve de de cette direction est donc omise. La preuve de la direction (3) $\Rightarrow$ (1) est immédiate puisqu'il existe toujours un LD-arbre équitable. Pour la preuve de la direction (2) $\Rightarrow$ (3) nous avons besoin d'un lemme. Pour une clause L nous notons $[L]$ l'ensemble des ses instances closes.

Lemme 4.2.1 *Soit P un programme logique, $\leftarrow A_1, \ldots, A_m$ un but, et $N_0, N_1, \ldots$ une LD-dérivation équitable infinie à partir de $P \cup \{\leftarrow A_1, \ldots, A_m\}$ avec séquence de substitutions $\theta_0, \theta_1, \ldots$. Alors pour tout $k \geq 0$ il existe un $n \geq 0$ tel que*

$$\bigcup_{i=1}^{m} [A_i(\theta_n \circ \ldots \circ \theta_0)] \subseteq T_P \downarrow k$$

Exercice 91
Démontrer le lemme 4.2.1.

Démonstration: (direction (2) $\Rightarrow$ (3) du théorème 4.2.2)

Supposons que $A \notin T_P \downarrow \omega$, et soit un LD-arbre équitable à partir de $P \cup \{\leftarrow A\}$. Par le lemme 4.2.1, toutes ses branches sont finies. Supposons que l'arbre contient un noeud de succès. Donc, il y a une LD-réfutation à partir de $P \cup \{\leftarrow A\}$, c'est-à-dire $A \in T_P \uparrow \omega \subseteq T_P \downarrow \omega$. Absurde. Il suit que l'arbre est un arbre fini d'échec. $\square$

Les résultat obtenues pour la négation en programmation logique sont résumées par la figure 4.5.

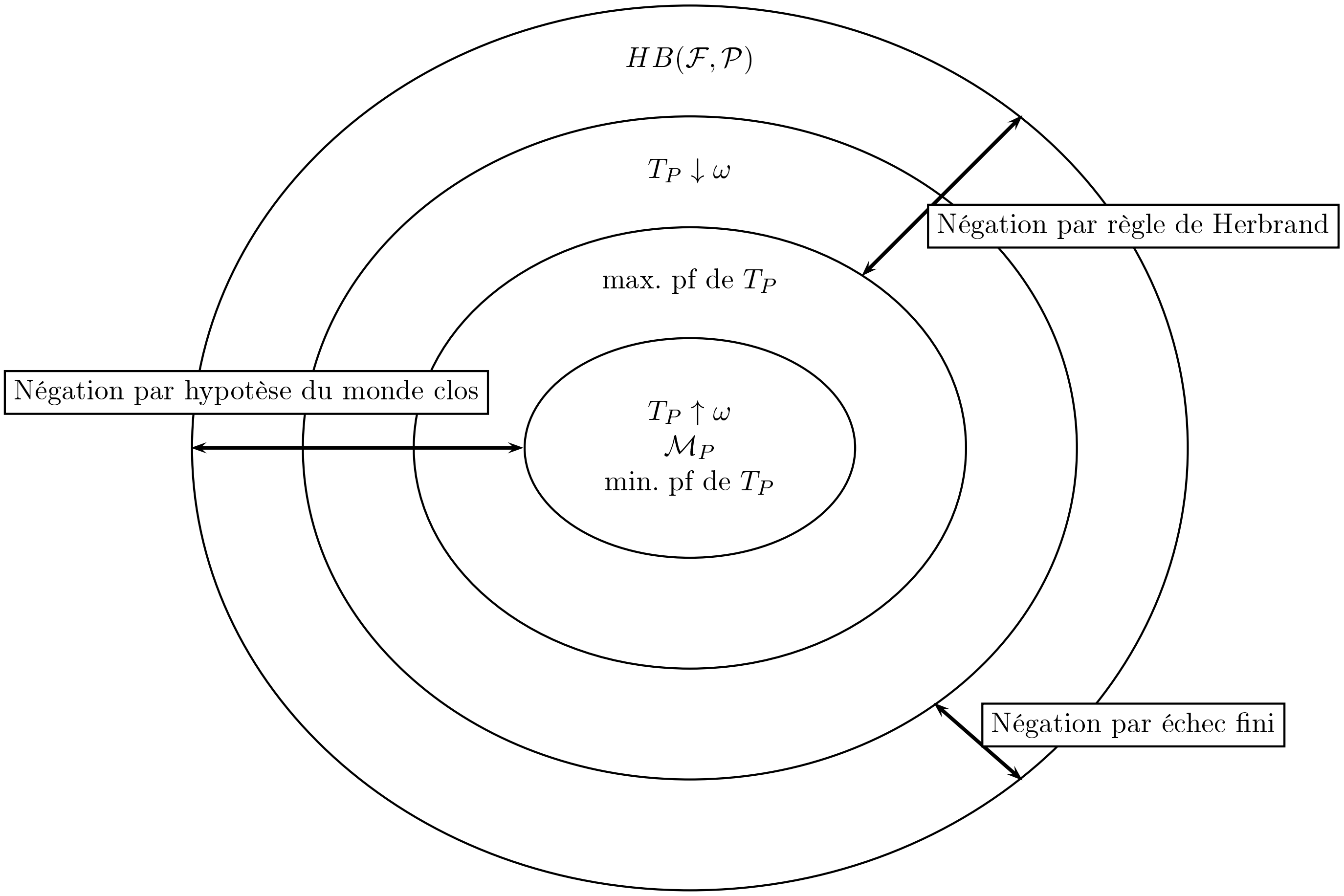

FIG. 4.5 – Résumé des résultats obtenues pour la négation en programmation logique.

4.2.3 Complétion de Clark

Nous avons vu qu'un littéral positif clos est dans la sémantique d'un programme logique si et seulement s'il est une conséquence logique du programme. Nous avons aussi vu qu'il n'est pas possible de donner une caractérisation équivalente de l'ensemble d'échec fini d'un programme logique par les conséquences logiques de P. L'objectif de cette section est de développer une théorie $comp(P)$ à partir de P telle que les littéraux clos (positives ou négatives) qui sont les conséquences logiques de $comp(P)$ sont exactement les atomes clos dans la sémantique du programme, respectivement les littéraux négatives dans l'ensemble d'échec fini du programme.

Étant donné un programme logique P, nous le transformons dans la formules $comp(P)$ en plusieurs étapes. Soit $=$ un nouveau symbole de relation. Pour tout symbole de prédicat p d'arité n soient $x_1^p, \ldots, x_n^p$ des nouvelles variables.

1. Transformer toute clause de la forme $p(t_1, \ldots, t_n) \leftarrow B_1, \ldots, B_m$, avec $\bar{y} = \mathrm{Var}(\leftarrow B_1, \ldots, B_m, t_1, \ldots, t_n)$ en

$$p(x_1^p, \ldots, x_n^p) \leftarrow \exists \bar{y}(x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge B_1 \wedge \ldots \wedge B_m)$$

2. Si l'ensemble des clauses obtenues dans l'étape précédente avec tête p est

$$\begin{array}{rl} p(x_1^p, \ldots, x_n^p) & \leftarrow F_1 \\ & \vdots \\ p(x_1^p, \ldots, x_n^p) & \leftarrow F_m \end{array}$$

alors nous les remplaçons par

$$\forall x_1^p, \ldots, x_n^p \big(p(x_1^p, \ldots, x_n^p) \leftrightarrow (F_1 \vee \ldots \vee F_m)\big)$$

ou il est entendu que la disjonction vide est définie comme $\perp$.

3. Ajouter les axiomes de l'égalité (voir exemple 3.1.7), plus les axiomes suivantes

$$\begin{array}{rl} & \{\tilde{\forall}(f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \ldots \wedge x_n = y_n) \mid f \in \mathcal{F}\} \\ \cup & \{\tilde{\forall}(f(x_1, \ldots, x_n) \neq g(y_1, \ldots, y_m) \mid f, g \in \mathcal{F}, f \neq g)\} \\ \cup & \{\tilde{\forall} x \neq t \mid x \in \mathrm{Var}(t), x \neq t\} \end{array}$$

Exemple 4.2.4 La complétion de Clark du programme de l'exemple 4.2.3 donne

$$\begin{array}{l} \forall x(Q(x) \leftrightarrow \exists y(x = a \wedge R(y))) \\ \forall x(R(x) \leftrightarrow \exists y(x = f(y) \wedge R(y))) \\ \forall x, y(f(x) = f(y) \rightarrow x = y) \\ \forall x(a \neq f(x)) \\ \{\forall x(x \neq f^n(x)) \mid n \geq 1\} \end{array}$$

plus les axiomes d'égalité.

Théorème 4.2.3 *Pour tout programme logique P et tout atome clos A*

$$P \models A \qquad ssi \qquad comp(P) \models A$$

Exercice 92
Démontrer le théorème 4.2.3.

Théorème 4.2.4 *Soit P un programme logique.*

1. *Une structure de Herbrand $\mathcal{A}$ est modèle de $comp(P)$ ssi elle est un point fixe de T_P.*
2. *Pour tous les atomes clos A, $comp(P) \cup \{A\}$ a un modèle de Herbrand ssi A est dans le point fixe le plus grand de T_P.*

Démonstration:

1. Toute structure de Herbrand satisfait les axiomes ajoutés à $comp(P)$ dans la troisième étape. L'équivalence suit par la définition de l'opérateur T_P.
2. Évidement, $comp(P) \cup \{A\}$ a un modèle de Herbrand ssi A est dans le modèle de Herbrand le plus grand de $comp(P)$. $\square$

Exemple 4.2.5 On reprend l'exemple 4.2.3. Par le théorème précédent, et parce que le point fixe le plus grand de ce programme est $\emptyset$, $comp(P) \cup \{A(a)\}$ n'a pas de modèle de Herbrand.

On peut donc définir une autre interprétation de la négation on programmation logique, appelée *Négation par la règle de Herbrand* en disant qu'un programme P implique un littéral négatif $\neg P(\bar{t})$ ssi $P \cup \{P(\bar{t})\}$ n'a pas de modèle de Herbrand.

Exercice 93
Montrer que $comp(P) \cup \{Q(a)\}$ a un modèle.

Théorème 4.2.5 *Soit P un programme logique et A un atome clos. Les énoncés suivants sont équivalents :*

1. *A est dans l'ensemble d'échec fini de P.*
2. *$A \notin T_P \downarrow \omega$*
3. *Tout LD-arbre équitable pour la requête $\leftarrow A$ est un arbre fini d'échec.*
4. *$comp(P) \models \neg A$*

Démonstration: Les équivalences entre (1), (2), (3) sont déjà données par le théorème 4.2.2.
Pour la direction (1) $\Rightarrow$ (4) : On peut montrer que, quand dans un LD-arbre on a un branchement de la forme

$$\begin{tikzcd}[column sep=small] & \leftarrow A_1, \ldots, A_n \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\ \leftarrow B_1^1, \ldots, B_{n_1}^1 & \ldots & \leftarrow B_1^k, \ldots, B_{n_k}^k \end{tikzcd}$$

alors

$$comp(P) \models \tilde{\exists}(A_1 \wedge \ldots \wedge A_n) \leftrightarrow \left[\tilde{\exists}(B_1^1 \wedge \ldots \wedge B_{n_1}^1) \vee \ldots \vee \tilde{\exists}(B_1^k \wedge \ldots \wedge B_{n_k}^k)\right]$$

Il suit, par induction, que dans un LD-arbre fini la requête initiale est équivalente la disjonction des requêtes qui correspondent aux feuilles. De plus, on obtient que la requête correspondante à un noeud d'échec est équivalente à une disjonction vide, c'est-à-dire à $\bot$. Donc, pour un LD-arbre d'échec avec requête initiale close A on obtient que

$$comp(P) \models A \leftrightarrow (\bot \vee \ldots \vee \bot)$$

Donc $comp(P) \models \neg A$.

La direction (4) $\Rightarrow$ (3) est omise. $\square$

Pour résumer, on a donc les caractérisations suivantes des différentes sémantiques de la négation : Pour un programme logique P et atome A,

$A \notin$ point fixe le plus petit de T_P	$\Leftrightarrow$	$P \not\models A$
$A \notin$ point fixe le plus grand de T_P	$\Leftrightarrow$	$comp(P) \cup \{A\}$ n'a pas de modèle de Herbrand
$A \notin T_P \downarrow \omega$	$\Leftrightarrow$	$comp(P) \models \neg A$.