



Answer Set Programming AnsProlog* – sémantique

Sémantique: traduction dans un formalisme mathématique

2 sémantiques:

1- **théorie des modèles – logique** (*un modèle est une interprétation qui satisfait une formule*)

2 – **sémantique du point fixe - procédurale**



Answer Set Programming AnsProlog* – sémantique

Sémantique: traduction dans un formalisme mathématique

2 sémantiques:

1- **théorie des modèles – logique** (*un modèle est une interprétation qui satisfait une formule*)

2 – **sémantique du point fixe - procédurale**

ASP – Sémantique: univers de Herbrand $HU_{\mathcal{L}}$

Un programme Π est un ensemble d'expressions ρ

$$\rho : L_0 \text{ or } L_1 \text{ or } \dots L_k \leftarrow L_{k+1}, L_{k+2}, \dots L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

Soit \mathcal{L} un langage contenant des

- Variables
- Fonctions
- Prédicats
- Connectives $\{\neg, \text{or}, \leftarrow, \text{not}, ',', '\cdot'\}$
- Ponctuations $\{', '(', ')', '\cdot'\}$
- Le symbole \perp

L'univers de Herbrand sur \mathcal{L} dénoté $HU_{\mathcal{L}}$, est l'ensemble des termes totalement instanciés formés avec les fonctions de \mathcal{L}

La base de Herbrand sur \mathcal{L} , dénotée $HB_{\mathcal{L}}$, est l'ensemble des atomes de \mathcal{L} totalement instanciés par des termes de $HU_{\mathcal{L}}$



ASP – Sémantique: instantiation des règles ρ

Un programme Π est un ensemble d'expression ρ

$$\rho : L_0 \text{ or } L_1 \text{ or } \dots L_k \leftarrow L_{k+1}, L_{k+2}, \dots L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

Soit une règle ρ donnée dans le langage \mathcal{L}

L' **instanciation** de ρ dans \mathcal{L} , notée $\text{ground}(\rho, \mathcal{L})$ est l' ensemble de toutes les règles obtenues en substituant les variables de par tous les éléments de $\text{HU}_{\mathcal{L}}$.

Exemple: soit les 3 règles $p(a) \leftarrow$. $p(b) \leftarrow$. $p(f(X)) \leftarrow p(X)$.

$\text{HU}_{\mathcal{L}}$ correspond à $\{a, b, f(a), f(b), f(f(a)), f(f(b)), f(f(f(a))), \dots\}$

$\text{ground}(p(f(X)) \leftarrow p(X), \mathcal{L})$ consiste dans les règles suivantes:

$p(f(a)) \leftarrow p(a)$. $p(f(b)) \leftarrow p(b)$. $p(f(f(a))) \leftarrow p(f(a))$. $p(f(f(b))) \leftarrow p(f(b))$.

$p(f(f(f(a)))) \leftarrow p(f(f(a)))$. $p(f(f(f(b)))) \leftarrow p(f(f(b)))$.

$p(f(f(f(f(a)))) \leftarrow p(f(f(f(a))))$





ASP: instantiation des programmes Π

Un programme Π est un ensemble d'expression ρ

$$\rho : L_0 \text{ or } L_1 \text{ or } \dots L_k \leftarrow L_{k+1}, L_{k+2}, \dots L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

**Pour n'importe quel programme Π on définit
l'instanciation du programme Π de la façon suivante:**

$$\text{ground}(\Pi, L) = \bigcup_{\rho \in \Pi} \text{ground}(\rho, L)$$

Instanciation d'un programme Π - $\text{ground}(\Pi, \mathcal{L})$

I
P

Programme Π

```
anc(X, Y) ← par(X, Y).  
anc(X, Y) ← par(X, Z),  
    anc(Z, Y).  
par(a, b) ← .  
par(b, c) ← .  
par(d, e) ← .
```

A

$\text{ground}(\Pi, \mathcal{L})$

$\text{anc}(a, a) \leftarrow \text{par}(a, a).$

$\text{anc}(a, b) \leftarrow \text{par}(a, b).$

$\text{anc}(a, c) \leftarrow \text{par}(a, c).$

$\text{anc}(a, c) \leftarrow \text{par}(a, d).$

...

$\text{anc}(a, a) \leftarrow \text{par}(a, a), \text{anc}(a, a).$

$\text{anc}(a, a) \leftarrow \text{par}(a, b), \text{anc}(b, a).$

...

$\text{par}(a, b) \leftarrow .$

$\text{par}(b, c) \leftarrow .$

$\text{par}(d, e) \leftarrow .$



Interprétation de Herbrand des programmes

Un programme Π de AnsProlog* est un ensemble d'expression ρ

$$\rho : L_0 \text{ or } L_1 \text{ or } \dots L_k \leftarrow L_{k+1}, L_{k+2}, \dots L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

Une interprétation partielle de Herbrand d'un programme AnsProlog* est n'importe quel sous-ensemble $I \subseteq HB_\Pi$

Une interprétation partielle S de I est dite satisfaire la règle ρ

$$\rho : L_0 \text{ or } L_1 \text{ or } \dots L_k \leftarrow L_{k+1}, L_{k+2}, \dots L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

Si

(i) $k=0$ et $L_0 = \perp$: $\{L_{k+1}, \dots, L_m\} \not\subseteq S$ ou $\{L_{m+1}, \dots, L_n\} \cap S \neq \emptyset$

(ii) sinon : $\{L_{k+1}, \dots, L_m\} \subseteq S$ et $\{L_{m+1}, \dots, L_n\} \cap S = \emptyset$ implique
 $\{L_0, \dots, L_k\} \cap S \neq \emptyset$





Modèles et Answer set

Un programme Π est un ensemble d'expression ρ

$$\rho : L_0 \text{ or } L_1 \text{ or } \dots L_k \leftarrow L_{k+1}, L_{k+2}, \dots L_m, \text{not } L_{m+1}, \dots, \text{not } L_n$$

Un modèle partiel de Herbrand A d'un programme Π est une interprétation partielle de Herbrand S de Π qui satisfait toutes les règles de Π et telle que si S contient une paire de littéraux complémentaires, alors S doit être égale à HB_{Π}

On dit alors que A est clos sous Π

Un answer set (ensemble réponse) d'un programme Π est un modèle partiel de Herbrand de Π qui est minimal – au sens de l'inclusion – parmi les modèles partiels de Herbrand de Π



En résumé

Sémantique des programmes AnsProlog*

- $\text{ground}(\Pi)$ rassemble l' ensemble des instances de Herbrand des règles ρ de Π
- Une *interprétation de Herbrand* d'un programme Π est un sous-ensemble I de la base de Herbrand de Π notée HB_{Π}
- Un *ensemble réponse* (« answer set ») d'un programme Π et un **sous-ensemble minimal S** de HB_{Π} qui est **clos** sous Π ; autrement dit, c'est un modèle partielle de Herbrand de Π



Exemples

- Exemple a

$a \leftarrow \text{not } b.$

Un modèle stable: $\{a\}$

- Exemple b

$a \leftarrow \text{not } b.$

$b \leftarrow \text{not } a.$

Deux modèles stables: $\{a\}$ et $\{b\}$



Exemples de programmes ASP

```
p ← a.  
q ← b.  
a ←.
```

Modèles: $\{a, b, q, p\}$,
 $\{a, p\}$, $\{a, p, q\}$

Modèles minimaux: $\{a, p\}$

Remarque: $\{a, b, q\}$ n'est pas
un modèle, $\{b, q\}$ non plus

```
anc(X, Y) ← par(X, Y).  
anc(X, Y) ← par(X, Z),  
    anc(Z, Y).  
par(a, b) ← .  
par(b, c) ← .  
par(d, e) ← .
```

Modèle minimal: $\{par(a, b)$,
 $par(b, c)$, $par(d, e)$,
 $anc(a, b)$, $anc(b, c)$,
 $anc(d, e)$, $anc(a, c)\}$

Remarque sur la non-monotonie

- *Usuellement, en logique pour tout couple de formules F et G toute interprétation X qui satisfait $F \wedge G$ satisfait aussi F*
Autrement dit, tout modèle de $F \wedge G$ est *a fortiori* un modèle de F
Ou, plus formellement si $X \models F \wedge G$ alors $X \models F$
C'est ce que l'on appelle la monotonie
- **Dans le cas des « ensembles de réponse » (Answer Sets) ce n'est plus vrai, car on s'intéresse aux modèles minimaux – la non monotonie est liée à la contrainte de minimalité**
Par exemple, $\{p, q\}$ est un ensemble réponse de $p \wedge q$, mais pas de p (car il ne serait pas minimal)
De même, $p \vee q$ possède deux ensembles de réponse, $\{p\}$ et $\{q\}$, tandis que $(p \vee q) \wedge (p \leftrightarrow q)$ en possède un seul, $\{p, q\}$.



Exemple de non monotonie: blocage

```
cours_MIA(N, M) :-  
    cours_MIA_prevu(N, M),  
    not exception_cours(N, M).  
  
cours_MIA(N, M) :-  
    rattrapage(N, M).  
  
exception_cours(N, M) :-  
    blocage(N, M),  
    not exception_blocage(N, M).  
  
exception_cours(N, M) :-  
    partiel(N, M),  
    not exception_partiel(N, M).  
  
exception_blocage(N, M) :-  
    salle_secours(N, M).  
  
exception_partiel(N, M) :-  
    rattrapage(N, M).
```

```
cours_MIA_prevu(15, mai).  
cours_MIA_prevu(24, avril).  
cours_MIA_prevu(3, avril).  
cours_MIA_prevu(27, mars).  
cours_MIA_prevu(20, mars).  
cours_MIA_prevu(13, mars).  
cours_MIA_prevu(6, mars).  
cours_MIA_prevu(27, fevrier).  
cours_MIA_prevu(20, fevrier).  
cours_MIA_prevu(13, fevrier).  
partiel(3, avril).  
rattrapage(3, avril).  
rattrapage(5, mai).  
blocage(20, mars).  
blocage(6, mars).  
salle_secours(20, mars).
```



Answer Set Programming AnsProlog* – sémantique

1- théorie des modèles – logique pour AnsProlog*

2 – sémantique du point fixe – procédurale

1. Pour les programmes AnsProlog^{not}

**2. Pour les programme AnsProlog - c'est-à-dire sans négation et
sans disjonction ($k=0$)**





Sémantique du point fixe AnsProlog^{-not}

—

point de vue calculatoire

- *Supposons que Π soit un ensemble, éventuellement infini, de clauses instanciées de AnsProlog^{-not} (c'est-à-dire sans not).*
- Soit 2^{HB_Π} l'ensemble de toutes les interprétations de Herbrand de Π .
- On définit l'opérateur $T_\Pi^0 : 2^{HB_\Pi} \rightarrow 2^{HB_\Pi}$
comme suit:

$$T_\Pi^0(I) = \left\{ L_0 \in HB_\Pi \middle| \begin{array}{l} \text{Π contient une règle $L_0 \leftarrow L_1, \dots, L_m$} \\ \text{telle que $\{L_1, \dots, L_m\} \subseteq I$} \end{array} \right\}$$

Cet opérateur est appelé l' *opérateur de conséquence immédiate*



Sémantique du point fixe – suite

Il convient d' itérer l'opérateur de conséquence immédiate

Pour cela on définit l' opérateur de conséquence à i pas: $T_{\Pi}^0 \uparrow i$

- Nous partons de $T_{\Pi}^0 \uparrow 0 = \{ \}$
- Puis on définit $T_{\Pi}^0 \uparrow (i + 1)$ comme $T_{\Pi}^0(T_{\Pi}^0 \uparrow i)$

On peut démontrer que pour toute base de Herbrand, même infinie, il existe un point fixe, c' est-à-dire une valeur de i au-delà de laquelle

$$T_{\Pi}^0 \uparrow (i + 1) = T_{\Pi}^0 \uparrow i = lfp(T_{\Pi}^0)$$



Sémantique du point fixe – suite

Pour tout programme Π de AnsProlog^{-not},
le modèle de Herbrand minimal de Π est unique

- Il correspond à l'ensemble réponse de Π (*answer set*)
- Il est égal à $lfp(T_\Pi^0)$
- Il est unique
- On le note $\mathcal{M}_0(\Pi)$

Exemple

`ground(Π, \mathcal{L})`

`anc(a, a) ← par(a, a).`

`anc(a, b) ← par(a, b).`

`anc(a, c) ← par(a, c).`

`anc(a, c) ← par(a, d)`

`anc(a, a) ← par(a, a), anc(a, a).`

`anc(a, a) ← par(a, b), anc(b, a)`

`par(a, b) ← .`

$T_{\Pi}^0 \uparrow 1 = \{ \text{par}(a,b), \text{par}(b,c), \text{par}(d,e) \}$

`par(b, c) ← .`

`par(d, e) ← .` $T_{\Pi}^0 \uparrow 2 = T_{\Pi}^0(T_{\Pi}^0 \uparrow 1) = \left\{ \begin{array}{l} \text{par}(a,b), \text{par}(b,c), \text{par}(d,e), \\ \text{anc}(a,b), \text{anc}(b,c), \text{anc}(d,e) \end{array} \right\}$

$T_{\Pi}^0 \uparrow 3 = T_{\Pi}^0(T_{\Pi}^0 \uparrow 2) = \left\{ \begin{array}{l} \text{par}(a,b), \text{par}(b,c), \text{par}(d,e), \\ \text{anc}(a,b), \text{anc}(b,c), \text{anc}(d,e), \text{anc}(a,c) \end{array} \right\}$

Programme Π

`anc(X, Y) ← par(X, Y).`

`anc(X, Y) ← par(X, Z), anc(Z, Y).`

`par(a, b) ← .`

`par(b, c) ← .`

`par(d, e) ← .`

$$T_{\Pi}^0 \uparrow 0 = \{ \}$$



Sémantique du point fixe AnsProlog

point de vue calculatoire

- Supposons que Π soit un ensemble, éventuellement infini, de clauses instanciées de AnsProlog (c'est-à-dire avec des not).
- Soit 2^{HB_Π} l'ensemble de toutes les interprétations de Herbrand de Π .
- Une première solution serait de définir l'opérateur $T_\Pi^1 : 2^{HB_\Pi} \rightarrow 2^{HB_\Pi}$ comme suit:

$$T_\Pi^1(I) = \left\{ L_0 \in HB_\Pi \middle| \begin{array}{l} \text{Π contient une règle } L_0 \leftarrow L_1, \dots, L_m, \text{not} L_{m+1}, \dots, \text{not} L_n \\ \text{telle que } \{L_1, \dots, L_m\} \subseteq I \quad \text{et} \quad \{L_{m+1}, \dots, L_n\} \cap I = \{\} \end{array} \right\}$$





Sémantique du point fixe AnsProlog

– Exemple

Considérons le programme Π

$a \leftarrow \text{not } b.$

$b \leftarrow \text{not } a.$

$$T_{\Pi}^1 \uparrow 0 = \{ \}$$

$$T_{\Pi}^1 \uparrow 1 = T_{\Pi}^1(T_{\Pi}^1 \uparrow 0) = \{a, b\}$$

$$T_{\Pi}^1 \uparrow 2 = T_{\Pi}^1(T_{\Pi}^1 \uparrow 1) = \{ \}$$

$$T_{\Pi}^1 \uparrow 3 = T_{\Pi}^1(T_{\Pi}^1 \uparrow 2) = \{a, b\}$$

- On constate une oscillation
- Il n'y a donc pas de convergence de la fonction vers un point fixe.
- En conséquence, il faut s'y prendre autrement....

Fin cours 4 –(5)



Sémantique du point fixe AnsProlog

Considérons le programme Π

Pour tout ensemble S d' atomes le programme Π^S est obtenu à partir de Π en détruisant:

- 1. Toute règle qui comporte un littéral $\text{not } L$ dans son corps avec $L \in S$**
- 2. Les littéraux de la forme $\text{not } L$ du corps des autres règles**

Π^S ne contient pas de not

On sait que Π^S admet un seul ensemble réponse, $\mathcal{M}_0(\Pi^S)$

Définition:

tout S tel que $S = \mathcal{M}_0(\Pi^S)$ est un ensemble réponse de Π

En conséquence, les ensembles réponses de Π (ou, modèles stables) sont $S1 = \{a\}$ et $S2 = \{b\}$

Sémantique du point fixe AnsProlog

Exemple

Considérons à nouveau le programme Π

$a \leftarrow \text{not } b.$
 $b \leftarrow \text{not } a.$

- Soit $S1 = \{a\}$, ce qui donne pour Π^{S1}
 $a \leftarrow .$
et $\mathcal{M}_0(\Pi^{S1}) = \{a\} = S1$
- Soit $S2 = \{b\}$, ce qui donne pour Π^{S2}
 $b \leftarrow .$
et $\mathcal{M}_0(\Pi^{S2}) = \{b\} = S2$
- Soit $S3 = \{a, b\}$, ce qui donne pour Π^{S3}

et $\mathcal{M}_0(\Pi^{S3}) = \{\}$
- Soit $S4 = \{ \}$ ce qui donne pour Π^{S4}
 $a \leftarrow .$
 $b \leftarrow .$
et $\mathcal{M}_0(\Pi^{S4}) = \{a, b\}$



Modèle stable – modèle minimal

- **Interprétation S de Herbrand d' un programme Π**
- **Modèle minimal (Sémantique logique):**
 - Interprétation de Herbrand close sous Π
 - Modèle minimal
- **Modèle stable (Sémantique du point fixe):**
 - L' ensemble Π^S est dérivé à partir des règles de Π
 - » Si une règle ρ de Π contient $\text{not}(L)$ avec $L \in S$, l'éliminer de Π^S
 - » Sinon, ajouter à Π^S les règles de Π dans lesquelles on a supprimé les négations de littéraux ($\text{not } L$)
 - Si $S = \mathcal{M}_0(\Pi^S)$ alors S est un modèle stable

