Correction du contrôle continu numéro 2
Partie modèles stables

Jean-François Baget `baget@lirmm.fr`

4 décembre 2024

Dans ce document, vous trouverez à la fois le sujet du contrôle continu (sur fond blanc), le corrigé lui-même (dans des boîtes jaunes), des versions alternatives du corrigé (dans des boîtes oranges) et de nombreux commentaires (dans des boîtes vertes telles que celle-ci). Si vous avez des questions, n'hésitez pas à me contacter par mail.
Un conseil: lisez ce document avec papier et stylo à la main, et refaites soigneusement les exercices en testant si vous arrivez au même résultat que ce que je vous propose. Tout particulièrement pour les exercices 1 à 3, les méthodes *doivent* être acquises.

1 Modèles Stables (version propositionnelle)

Dans cette section, nous ne considérons que des programmes et des règles propositionnelles. Les définitions et notations utiles pour les exercice suivants sont rappelées en annexe (faites attention à la définition du corps négatif qui, dans le cas propositionnel, est ici simplifiée). Pour les exercices indiqués par une $*$, plus difficiles, des points bonus seront accordés suivant la qualité de la justification.

Dans l'ensemble, je suis assez déçu par vos copies, et considère qu'il y a un réel manque de travail. Les statistiques suivantes sont éloquentes (je rappelle que cette partie était notée sur 7 points): moyenne: 2.62, écart type: 2.20, médiane: 2.25.

Il y a quand même 3 étudiants qui ont obtenu la note maximale: bravo à eux. D'autre part, 6 étudiants ont eu strictement plus de 4 points, et je considère ces copies comme très correctes: continuez dans cette voie. Pour les 10 étudiants qui ont moins de 1: avez vous relu le cours ?

1.1 Utilisation de la définition par point fixe

Les 3 premiers exercices étaient notés sur un total de 3 pts. Ils ne nécessitent que l'application d'un algorithme simple, vu plusieurs fois en cours, et sur lequel j'avais insisté, car 1) il est nécessaire pour comprendre la suite du cours et 2) il tombe chaque fois à l'examen. Il est donc très inquiétant que plus de la moitié des étudiants n'ait pas bien réussi cette partie.

Soit Π le programme propositionnel suivant.

$$
\begin{array}{ccc}
 & b, not\, c \to d & d, not\, f \to f \\
a & & \\
a \to b & b, not\, d \to c & c, not\, g \to h
\end{array}
$$

Exercice 1 Utiliser la méthode du point fixe pour savoir si $E_1 = \{a, b, c, h\}$ est un modèle stable de Π. Vous donnerez explicitement le programme réduit pour justifier votre réponse.

En utilisant l'algorithme vu en cours, nous construisons le programme réduit $\Pi_{|E_1}$ avec $E_1 = \{a, b, c, h\}$ de la façon suivante:

Π	$\Pi_{\|\mathbf{E_1}}$	**Justification**
a	a	règle positive
$a \to b$	$a \to b$	règle positive
$b, not\, c \to d$		$c \in E_1$
$b, not\, d \to c$	$b \to c$	$d \notin E_1$
$d, not\, f \to f$	$d \to f$	$f \notin E_1$
$c, not\, g \to h$	$c \to h$	$g \notin E_1$

La saturation du programme positif $\Pi_{|E_1}$ nous donne l'ensemble d'atomes $(\Pi_{|E_1})^* = \{a, b, c, h\}$. Puisque $E_1 = (\Pi_{|E_1})^*$, alors par définition **$\mathbf{E_1}$ est un modèle stable de $\mathbf{\Pi}$**.

Construction du programme réduit: j'ai utilisé ici la définition *faible* du programme réduit (celle qui était rappelée en annexe), où nous n'examinons que les corps négatifs. Vous auriez pu également utiliser la version *forte*, qui examine *également* les corps positifs (dans 1 ou 2 copies, seuls les corps positifs étaient examinés, ce qui donne n'importe quoi). Avec la définition forte:

Π	$\Pi_{\|\mathbf{E_1}}$	**Justification**
a	a	fait
$a \to b$	$a \to b$	règle positive et $a \in E_1$
$b, not\, c \to d$		$c \in E_1$
$b, not\, d \to c$	$b \to c$	$d \notin E_1$ et $b \in E_1$
$d, not\, f \to f$		$d \notin E_1$
$c, not\, g \to h$	$c \to h$	$g \notin E_1$ et $c \in E_1$

La définition forte donne un programme réduit plus petit que la définition faible, mais on a la propriété suivante: *soit* Π *un programme et* E *un ensemble d'atomes. Alors* $E = (\Pi^+_{|E})^*$ si et seulement si $E = (\Pi^-_{|E})^*$ (où $\Pi^+_{|E}$ est le programme construit avec la version forte et $\Pi^-_{|E}$ celui avec la version faible). Il est donc équivalent de tester avec la version forte ou la version faible du programme réduit. Par contre, il est faux de dire que les programmes générés par ces deux versions ont toujours la même saturation (comme j'ai du l'affirmer pendant le cours). Voir exercice 3.

Remarques en vrac:

- je n'ai pas détaillé dans mon corrigé comment on obtient la saturation. Je suppose que c'est un acquis. Pourtant, plusieurs étudiants ont correctement calculé le programme réduit, mais l'ont mal saturé. Soyez un peu plus rigoureux.
- dans quelques copies, on voit dans le programme réduit des règles qui ne sont pas positives (il reste des *not*). Or *le programme réduit est toujours positif.* La méthode utilisée est donc fausse. Mais ce qui me choque le plus, c'est qu'ils arrivent quand même à calculer une saturation: *la saturation n'est définie que pour les programmes positifs.* Sinon, on peut considérer des *bonnes dérivations*, mais leur résultat n'est pas unique et ça va donc être compliqué de tester l'égalité pour la définition par point fixe.
- beaucoup trop d'étudiants ont préféré, pour une raison ou une autre, éviter le calcul du programme réduit. C'est pourtant ce qui était demandé. Dans tous les cas (sauf 1), ce calcul a été remplacé par des justifications plutôt vagues. Avec ce qui avait été vu en cours (avant l'algorithme ASPERIX), la seule façon de faire était d'exhiber une dérivation:

$$\{a\} \xrightarrow{a \to b} \{a, b\} \xrightarrow{b, not\, d \to c} \{a, b, c\} \xrightarrow{c, not\, g \to h} \{a, b, c, h\}$$

puis de justifier que cette dérivation était persistante (les règles appliquées sont toujours applicables sur le résultat $\{a, b, c, h\}$) et complète (aucune autre règle n'est applicable sur le résultat) pour affirmer (par théorème vu en cours) que le résultat de cette bonne dérivation est bien un modèle stable. Mais ce n'est pas ce qui était demandé.

- l'algorithme demandé ne nécessite que des définitions de base de théorie des ensembles. En M2, il serait temps de faire la différence entre l'inclusion ($\subseteq$) et l'appartenance ($\in$), et de maitriser les opérations d'intersection ($\cap$) et d'union ($\cup$).
- trop souvent, j'ai vu le calcul du programme réduit, le calcul du saturé, et puis c'est tout. Prenez l'habitude de conclure, la question étant "est-ce un modèle stable", la dernière ligne de votre réponse doit être "c'est un modèle stable" ou "ce n'est pas un modèle stable".

Exercice 2 Même question avec $E_2 = \{a, b, d\}$.

En utilisant l'algorithme vu en cours, nous construisons le programme réduit $\Pi_{|E_2}$ avec $E_2 = \{a, b, d\}$ de la façon suivante:

Π	$\Pi_{\|E_2}$	Justification
a	a	règle positive
$a \to b$	$a \to b$	règle positive
$b, not\, c \to d$	$b \to d$	$c \notin E_2$
$b, not\, d \to c$		$d \in E_2$
$d, not\, f \to f$	$d \to f$	$f \notin E_2$
$c, not\, g \to h$	$c \to h$	$g \notin E_2$

La saturation du programme positif $\Pi_{|E_2}$ nous donne l'ensemble d'atomes $(\Pi_{|E_2})^* = \{a, b, d, f\}$. Puisque $E_2 \neq (\Pi_{|E_2})^*$, alors par définition **E_2 n'est pas un modèle stable de Π**.

Mêmes remarques que pour l'exercice précédent. Je vous donne quand même le programme construit avec la version forte de la définition du programme réduit:

Π	$\Pi_{\|E_2}$	Justification
a	a	fait
$a \to b$	$a \to b$	règle positive et $a \in E_2$
$b, not\, c \to d$	$b \to d$	$c \notin E_2$ et $b \in E_2$
$b, not\, d \to c$		$d \in E_2$
$d, not\, f \to f$	$d \to f$	$f \notin E_2$ et $d \in E_2$
$c, not\, g \to h$		$c \notin E_2$

On ne peut pas prouver que E n'est pas un modèle stable en exhibant une unique bonne dérivation. Il faudrait prouver qu'aucune bonne dérivation n'a E pour résultat, en les exhibant toutes.

Exercice 3 Même question avec $E_3 = \{a, g\}$. Pour cet exercice, vous pourrez éviter le calcul du programme réduit par une remarque judicieuse.

En l'absence de remarque judicieuse, il était tout à fait possible de refaire (encore) un programme réduit. C'est la **version 1** de ma correction. On pouvait aussi remarquer que g ne peut pas apparaitre dans un modèle stable, ce qui pouvait donner un indice pour l'exercice 4. C'est la **version 2** de ma correction. Enfin, on pouvait remarquer que b devrait apparaitre dans tout modèle stable, ce qui donnait un indice pour l'exercice 5. C'est la **version 3** de ma correction. Une seule des 3 versions était demandée.

Exercice 3, version 1: En utilisant l'algorithme vu en cours, nous construisons le programme réduit $\Pi_{|E_3}$ avec $E_3 = \{a, g\}$ de la façon suivante:

Π	$\Pi_{\|\mathbf{E_2}}$	**Justification**
a	a	règle positive
$a \rightarrow b$	$a \rightarrow b$	règle positive
$b, not\, c \rightarrow d$	$b \rightarrow d$	$c \notin E_2$
$b, not\, d \rightarrow c$	$b \rightarrow c$	$d \notin E_2$
$d, not\, f \rightarrow f$	$d \rightarrow f$	$f \notin E_2$
$c, not\, g \rightarrow h$		$g \in E_2$

La saturation du programme positif $\Pi_{|E_3}$ nous donne l'ensemble d'atomes $(\Pi_{|E_3})^* = \{a, b, c, d, f\}$. Puisque $E_3 \neq (\Pi_{|E_3})^*$, alors par définition **E_3 n'est pas un modèle stable de Π**.

La aussi, nous aurions pu utiliser la version forte de la définition du programme réduit.

Π	$\Pi_{\|\mathbf{E_2}}$	**Justification**
a	a	fait
$a \rightarrow b$	$a \rightarrow b$	règle positive et $a \in E_3$
$b, not\, c \rightarrow d$		$b \notin E_3$
$b, not\, d \rightarrow c$	$b \rightarrow c$	$b \notin E_3$
$d, not\, f \rightarrow f$		$d \notin E_3$
$c, not\, g \rightarrow h$		$g \in E_3$

Remarquez ici que la saturation du programme génère $\{a, b, c\}$, ce qui est différent de la saturation générée avec la version faible du programme réduit $\{a, b, c, d, f\}$. Voir commentaire de l'exercice 1. On peut ici vérifier également avec cette version du programme réduit que E_3 n'est pas un modèle stable, puisque $\{a, b, c\} \neq E_3$.

Remarque: très peu de personnes ont utilisé le programme réduit. Ce qui est dommage puisque les justifications apportées étaient souvent très vagues.

Exercice 3, version 2: Soit E un modèle stable quelconque de Π. Puisque l'atome g n'apparait dans aucune tête de règle de Π, par construction g n'apparait pas non plus dans la tête d'aucune règle de $\Pi_{|E}$. Donc $g \notin (\Pi_{|E})^*$. Et comme E est par hypothèse un modèle stable, on a par définition $E = (\Pi_{|E})^*$, et donc $g \notin E$. Nous avons prouvé que g ne pouvait apparaitre dans aucun modèle stable de Π, et comme E_3 contient g, il s'ensuit que **E_3 n'est pas un modèle stable de Π**.

L'intérêt d'avoir une justification rigoureuse dans le cas particulier de Π et E_3 est que nous allons pouvoir aisément l'adapter à une démonstration générale pour l'exercice 4. En effet, la seule particularité de g que nous avons utilisée ici est qu'il n'apparait dans aucune tête de règle.

Exercice 3, version 3: Soit E un modèle stable quelconque de Π. Puisque a et $a \rightarrow b$ sont des règles positives, elle apparaissent nécessairement dans le programme réduit $\Pi_{|E}$. Ainsi, la saturation génèrera un ensemble d'atomes $(\Pi_{|E})^*$ contenant $\{a, b\}$. Et comme E est par hypothèse un modèle stable, on a par définition $E = (\Pi_{|E})^*$, et donc $\{a, b\} \subseteq E$.

Nous avons prouvé que $\{a, b\}$ était inclus dans tous les modèles stables de Π, et comme E_3 ne contient pas b, il s'ensuit que **E_3 n'est pas un modèle stable de Π**.

Ici aussi, nous pourrons facilement adapter notre justification particulière au cas général pour répondre à la question 5: la seule particularité de $\{a, b\}$ que nous avons utilisée est que c'est la saturation par les règles positives du programme.

1.2 Génération de tous les modèles stables

Un algorithme immédiat pour générer tous les modèles stables d'un programme Π est de calculer son *vocabulaire* V_Π (c'est à dire l'ensemble des atomes apparaissant dans une règle de Π), de générer chaque sous-ensemble E de V_Π, puis de tester E par la définition par point fixe. Cet algorithme fonctionne en temps $2^{|V_\Pi|} \times poly(|\Pi|)$. Il est donc urgent de réduire le nombre de sous-ensembles candidats à tester. Dans les exercices suivants, nous allons calculer deux sous-ensembles V_Π^{min} et V_Π^{max} de V_Π qui ont les propriétés suivantes:

- si $a \in V_\Pi^{min}$, alors a est dans tous les modèles stables de Π
- si $a \notin V_\Pi^{max}$, alors a n'est dans aucun modèle stable de Π

Ainsi, il sera uniquement nécessaire de tester tous les sous-ensembles de V_Π^{max} contenant V_Π^{min} pour retrouver tous les modèles stables de Π.

Exercice 4* Trouvez une méthode pour calculer un V_Π^{max} (qui sera, pour certains programmes, strictement inclus dans V_Π). *Indice:* revoyez l'exercice 3. Vous justifierez votre réponse.

Nous allons donner plusieurs idées, de plus en plus efficaces, pour construire V_Π^{max}.

Exercice 4, version 1: Nous allons prouver la proposition suivante:

Proposition 1.1: *Soit* Π *un programme. Un atome est dit* générable *par* Π *si il apparait dans la tête d'une règle de* Π. *Alors aucun modèle stable de* Π *ne contient d'atome non générable par* Π.

On peut donc **définir $\mathbf{V_\Pi^{max}}$ comme l'ensemble des atomes générables par $\mathbf{\Pi}$**. Dans notre exemple, on a $V_\Pi^{max} = V_\Pi \setminus \{g\}$, il y a donc bien des cas où notre calcul donne $V_\Pi^{max} \subsetneq V_\Pi$. Il reste donc à démontrer la proposition:

Preuve: soit a un atome non générable, et soit E un modèle stable quelconque d'un programme Π. Puisque l'atome a n'apparait dans aucune tête de règle de Π, par construction a n'apparait pas non plus dans la tête d'aucune règle de $\Pi_{|E}$. Donc $a \notin (\Pi_{|E})^*$. Et comme E est par hypothèse un modèle stable, on a par définition $E = (\Pi_{|E})^*$, et donc $a \notin E$. □

La version 1 est plus ou moins l'idée qu'ont eu la plupart des étudiants qui ont abordé cet exercice, avec plus ou moins de précision et de simplicité dans son énoncé, et très rarement avec une justification/preuve satisfaisante. On peut faire un peu mieux, si on considère l'exemple suivant:

$$a \qquad\qquad a, not\, b \to c \qquad\qquad d, not\, b \to e$$

Ici, notre version 1 donne $V_\Pi^{max} = \{a, c, e\}$. Pourtant, on se demande bien comment fera un programme réduit quelconque pour générer e. Il y a bien une saturation à calculer quelque part, c'est ce qu'on va faire dans la version 2.

Exercice 4, version 2: Nous allons prouver la proposition suivante:

Proposition 1.2: *Soit* Π *un programme. Notons* $pos(\Pi) = \{pos(R) \mid R \in \Pi\}$ *l'ensemble des règles de* Π *auxquelles on a enlevé le corps négatif. Un atome est dit* générable *par* Π *si il apparait dans* $pos(\Pi)^*$. *Alors aucun modèle stable de* Π *ne contient d'atome non générable par* Π.

On peut donc **définir $\mathbf{V_\Pi^{max}}$ comme l'ensemble des atomes générables par $\mathbf{\Pi}$**. Dans notre exemple, on a $V_\Pi^{max} = V_\Pi \setminus \{g\}$, il y a donc bien des cas où notre calcul donne $V_\Pi^{max} \subsetneq V_\Pi$. Il reste donc à démontrer la proposition:

Preuve: Soit E un modèle stable quelconque d'un programme Π. Nous remarquons tout d'abord que $pos(\Pi) = \Pi_{|\emptyset}$ (avec la définition *faible* du programme réduit). Comme $\emptyset \subseteq E$, on voit, par construction, que $\Pi_{|E} \subseteq \Pi_{|\emptyset}$, et donc que $E = (\Pi_{|E})^* \subseteq (\Pi_{|\emptyset})^* = pos(\Pi)^*$.

Soit maintenant un atome a non générable quelconque, c'est à dire $a \notin pos(\Pi)^*$. Comme $E \subseteq pos(\Pi)^*$, alors $a \notin E$: a n'appartient à aucun modèle stable de Π. □

Je crois que seuls 2 ou 3 étudiants ont donné la version 2, avec une démonstration un peu différente de celle donnée ci-dessus. Mais on aurait pu faire encore mieux! Observons l'exemple suivant:

$$a \qquad\qquad a \to b \qquad\qquad a, not\, b \to c$$

Avec la version 2 (comme avec la version 1), on trouve $V_\Pi^{max} = \{a, b, c\}$. Mais on se demande bien comment on pourrait générer c, puisque b sera dans tous les modèles stables, et donc la règle $a \to c$ ne sera dans aucun programme réduit dont le saturé est un modèle stable. L'idée est maintenant la suivante: si on a déjà calculé V_Π^{min}, on pourra calculer un meilleur V_Π^{max}. Utilisons maintenant cette idée dans la version 3.

Exercice 4, version 3: Nous allons prouver la proposition suivante:

Proposition 1.3: *Soit* Π *un programme, dont nous avons calculé* V_Π^{min}. *Un atome est dit* générable *par* Π *si il appartient à* $(\Pi_{|V_\Pi^{min}})^*$. *Alors aucun modèle stable de* Π *ne contient d'atome non générable par* Π.

On peut donc **définir $\mathbf{V_\Pi^{max}}$ comme l'ensemble des atomes générables par $\mathbf{\Pi}$**. Dans notre exemple, on a $V_\Pi^{max} = V_\Pi \setminus \{g\}$, il y a donc bien des cas où notre calcul donne $V_\Pi^{max} \subsetneq V_\Pi$. Il reste donc à démontrer la proposition:

Preuve: Soit E un modèle stable quelconque d'un programme Π dont on a calculé $V_\Pi^{\min}$. On a $V_\Pi^{\min} \subseteq E$ (avec la définition *faible* du programme réduit), donc $\Pi_{|E} \subseteq \Pi_{|V_\Pi^{\min}}$, et donc $E = (\Pi_{|E})^* \subseteq (\Pi_{|V_\Pi^{\min}})^*$.

Soit maintenant un atome a non générable quelconque, c'est à dire $a \notin (\Pi_{|V_\Pi^{\min}})^*$. Comme $E \subseteq (\Pi_{|V_\Pi^{\min}})^*$, alors $a \notin E$: a n'appartient à aucun modèle stable de Π. □

Exercice 5∗ ChatGPT m'a proposé pour calculer V_Π^{min} la notion d'*ensemble garanti*, c'est à dire l'ensemble des faits apparaissant dans le programme Π. Il m'a même (correctement) démontré que tous les atomes de l'ensemble garanti sont dans tous les modèles stables de Π. Proposez-moi pour V_Π^{min} un ensemble possiblement plus grand que l'ensemble garanti utilisé par ChatGPT. *Indice:* vous pouvez penser à la règle $a \to b$ du programme précédent. Vous justifierez soigneusement votre réponse, par une démonstration rigoureuse.

Là encore, nous allons proposer plusieurs idées, de plus en plus efficaces, pour calculer $V_\Pi^{\min}$

Exercice 5, version 1: Nous allons prouver la proposition suivante:

Proposition 2.1: *Soit* Π *un programme. Un atome* a *est dit* garanti *par* Π *si il appartient à* $(\Pi^+)^*$, *où* Π^+ *est la partie positive de* Π, *c'est à dire le sous-ensemble de* Π *composé de ses règles positives. Alors un atome garanti par* Π *est dans tous les modèles stables de* Π.

On peut donc **définir $\mathbf{V_\Pi^{min}}$ comme l'ensemble des atomes garantis par $\mathbf{\Pi}$**. Dans notre

exemple, on a $V_\Pi^{min} = \{a, b\}$, ce qui est plus grand que l'ensemble $\{a\}$ garanti par ChatGPT. Il reste donc à démontrer la proposition:

Preuve: Soit E un modèle stable quelconque d'un programme Π. Voir que $\Pi_{|E}$, par construction, contient toutes les règles positives de Π. On a donc $\Pi^+ \subseteq \Pi_E$ et donc $(\Pi^+)^* \subseteq (\Pi_{|E})^* = E$. Soit maintenant un atome garanti a quelconque, c'est à dire un élément de $(\Pi^+)^*$. C'est aussi un élément du modèle stable quelconque E, ainsi a appartient à tous les modèles stables de E. □

Beaucoup de ceux qui ont répondu à cette question ont imaginé une version située quelque part entre la version 0 de ChatGPT (seuls les faits sont garantis) et la version 1 définie ci-dessus (les atomes générés par saturation de la partie positive du programme sont garantis). Cette "version 0.5" pourrait être décrite de la façon suivante (ça a été fait avec plus ou moins d'exactitude):

- les faits sont garantis;
- si une règle R est telle que $body^+(R) \subseteq facts(\Pi)$, et $body^-(R) = \emptyset$, alors $head(R)$ est garanti.

C'est dommage, car ça ressemble beaucoup à la saturation par les règles positives, mais c'est juste de la saturation au rang 1. J'ai quand même donné les points, car ça répondait à mes spécifications (faire mieux que ChatGPT).

D'autres ont amélioré cette version en essayant (ce qui n'a pas été bien fait) de redéfinir la saturation (mais pourquoi s'embêter comme ça?). Je rappelle ici comment on définit mathématiquement la saturation d'un programme positif (ce qui ressemble à ce qu'ont essayé de faire certains):

Définition (saturation d'un programme positif): *soit* Π *un programme positif. Alors la saturation de* Π *est le plus petit ensemble d'atomes* Π^* *tel que, si* R *est une règle de* Π *et* $body^+(R) \subseteq \Pi^*$*, alors* $head(R) \in \Pi^*$*.*

Notons que, comme souvent au cours de cette correction, je n'ai pas eu besoin de faire un cas particulier pour les faits: leur corps positif vide est toujours inclus dans Π^*, et donc leur tête doit toujours appartenir à Π^*.

Enfin, si on veut améliorer notre calcul de $V_\Pi^{\min}$, il faudrait saturer avec un ensemble de règles plus grand que juste Π^+. Mais pour pouvoir assurer qu'une autre règle ne sera pas bloquée, il faudra être certain qu'aucun des éléments du corps négatif ne sera dans un modèle stable, c'est à dire que tous les éléments du corps négatif seront dans $V_\Pi \setminus V_\Pi^{\max}$ (un peu l'idée de la version 3 de l'exercice 4). Pour le dire autrement, si aucun des atomes du corps négatif n'est générable, alors on est certain que la règle ne sera pas bloquée, et sa version positive sera systématiquement dans le programme réduit de tous les modèles stables. Ce n'est donc pas Π^+ qu'il faut saturer, mais $\Pi_{|(V_\Pi \setminus V_\Pi^{\max})}$. Développons donc cette idée dans la version 2.

Exercice 5, version 2: Nous allons prouver la proposition suivante:

Proposition 2.2: *Soit* Π *un programme, dont on a calculé* V_Π^{max}*. Un atome* a *est dit* garanti *par* Π *si il appartient à* $(\Pi_{|(V_\Pi \setminus V_\Pi^{max})})^*$*. Un atome garanti par* Π *est dans tous les modèles stables de* Π*.*

On peut donc **définir $\mathbf{V_\Pi^{min}}$ comme l'ensemble des atomes garantis par $\mathbf{\Pi}$**. Dans notre exemple, on a $V_\Pi^{min} = \{a, b\}$, ce qui est plus grand que l'ensemble $\{a\}$ garanti par ChatGPT. Il reste donc à démontrer la proposition:

Preuve: Soit E un modèle stable quelconque d'un programme Π, pour lequel on a déjà calculé $V_\Pi^{\max}$. Voir que $\Pi_{|E}$, par construction, contient les versions positives de toutes les règles R dont aucun atome de $body^-(R)$ n'est dans E. En particulier, il contient les versions positives de toutes les règles R dont tous les atomes de $body^-(R)$ sont dans $V_\Pi \setminus V_\Pi^{\max}$. On a donc $\Pi_{|(V_\Pi \setminus V_\Pi^{\max})} \subseteq \Pi_E$

et donc $(\Pi_{|(V_\Pi \setminus V_\Pi^{\max})})^* \subseteq (\Pi_{|E})^* = E$. Soit maintenant un atome garanti a quelconque, c'est à dire un élément de $(\Pi_{|(V_\Pi \setminus V_\Pi^{\max})})^*$. C'est aussi un élément du modèle stable quelconque E, ainsi a appartient à tous les modèles stables de E. □

Au cours de ces deux exercices, nous avons vu différentes versions du calcul de $V_\Pi^{\min}$ et de $V_\Pi^{\max}$: 3 versions de $V_\Pi^{\max}$ qui calculent des ensembles de plus en plus petits, et 2 versions de $V_\Pi^{\min}$ qui calculent des ensembles de plus en plus grand. Les dernières versions de ces 2 calculs utilisent le résultat de l'autre, avec la propriété suivante: plus le $V_\Pi^{\min}$ donné comme paramètre de $V_\Pi^{\max}$ (V3) est grand, plus le résultat sera petit; et plus le $V_\Pi^{\max}$ donné comme paramètre de $V_\Pi^{\min}$ (V2) est petit, plus le résultat sera grand. On voit donc qu'il pourrait être intéressant d'améliorer itérativement le calcul des $V_\Pi^{\min}$ et $V_\Pi^{\max}$, en se servant des améliorations successives jusqu'à stabilité. On pourrait le faire, par exemple, avec le petit programme python suivant (qui pourrait être amélioré, mais je voulais rester simple):

```
def compute_v_sets(program):
  v_min = compute_vmin_v1(program)
  v_max = compute_vmax_v2(program)
  change = True
  while(change):
      v_min_aux = compute_vmin_v2(program, v_max)
      v_max_aux = compute_vmax_v3(program, v_min)
      if len(v_min_aux) == len(v_min) and len(v_max_aux) == len(v_max):
          change = False
      else:
          v_min = v_min_aux
          v_max = v_max_aux
  return v_min, v_max
```

Annexe A: rappels de cours, et notations

En version propositionnelle, une *règle* est un objet R de la forme

$$p_1, \ldots, p_k, not\, n_1, \ldots, not\, n_q \rightarrow h$$

où $h, p_1, \ldots, p_k, n_1, \ldots, n_q$ sont des atomes propositionnels. On note $head(R) = h$, $body^+(R) = \{p_1, \ldots, p_k\}$ et $body^-(R) = \{n_1, \ldots, n_q\}$. Une règle R est un *fait* lorsque $body^+(R) = body^-(R) = \emptyset$ (et dans ce cas on la note simplement h), et elle est dite *positive* lorsque $body^-(R) = \emptyset$. La *partie positive* d'une règle R est la règle $pos(R) = p_1, \ldots, p_k \rightarrow h$ obtenue en enlevant tous les atomes du corps négatif.

Un programme Π est un ensemble de règles. Un programme sera dit *positif* lorsque toutes ses règles sont positives. La *partie positive* de Π est le programme (positif) Π^+ constitué de toutes les règles positives de Π. Si Π est un programme positif, la *saturation* de Π est l'ensemble d'atomes Π^* obtenu par une dérivation complète de Π (c'est donc son *modèle canonique*).

Soit Π un programme, et E un ensemble d'atomes, alors le *programme* Π *réduit par* E, noté $\Pi_{|E}$, est l'ensemble de règles obtenu de la façon suivante: pour chaque règle R de Π, si $body^-(R) \cap E = \emptyset$, alors $\Pi_{|E}$ contient $pos(R)$, sinon la règle est absente de $\Pi_{|E}$. Un *modèle stable* de Π est un ensemble d'atomes E tel que $(\Pi_{|E})^* = E$ (c'est la *définition par point fixe*).