Sémantique des langages de programmation
Master2, 1ère leçon
Sémantique naturelle et preuves

Yves Bertot

November 5, 2004

1 Sémantique naturelle

1.1 Le langage que nous étudions

Nous allons décrire un petit langage impératif, déjà utilisé dans les livres de Nielson&Nielson [1] et G. Winskel [2]. La syntaxe de ce langage est constituée de la manière suivante:

- Chaque programme est constitué d'une liste de déclarations et d'une instruction.
- Les instructions sont de quatre types différents: `skip` est une instruction qui ne fait rien; x `:=` e représente l'affectation, i_1`;`i_2 représente la séquence de deux instructions, `while` b `do` i représente une boucle. Nous utiliserons les variables i, i_0, i', i_a pour représenter des instructions.
- les expressions arithmétiques sont soit des variables x, y, soit des entiers relatifs, soit l'addition de deux expressions e_1 `+` e_2. Nous utilserons les variables e, e_1, e' comme méta-variables pour représenter des expressions.
- les expressions booléenes sont toujours des comparaisons de deux expressions arithmétiques e_1 `<` e_2. Nous utiliserons des variables b, b', pour représenter les expressions booléennes.

Il est bien sûr possible d'ajouter plus d'opérateurs parmi les expressions arithmétiques, plus d'opérateurs parmi les expressions booléennes et d'autoriser l'utilisation de variables booléennes. Il est aussi pertinent de munir le langage d'expressions conditionnelles "si ... alors ... sinon", mais ceci n'est pas très important pour les besoins du cours car l'instruction `while` suffit pour montrer comment définir un langage de programmation.

1.2 Quelques notations

Dans l'exécution d'un programme, toute expression représente une valeur entière que l'on veut calculer et toute expression booléenne représente une valeur booléenne. Ces expressions peuvent contenir des variables et l'expression n'a donc de sens que dans un certain contexte. Dans la suite, nous utiliserons toujours le terme *environnement* pour décrire un tel contexte. Un environnement sera une liste finie de couples *variables-valeur*. Nous utiliserons les variables ρ, ρ' pour dénoter les environnements, nous noterons $\emptyset$ l'environnement vide et $(x, v) \cdot \rho$, l'environnement qui est obtenu en ajoutant l'association de x à v à l'environnement ρ. Par exemple, dans l'environnement $(x, 3) \cdot (y, 5) \cdot \emptyset$, l'expression x+y doit avoir la valeur 8.

Notre objectif est de décrire cet aspect de façon formelle, sans qu'il y ait de doute possible. Pour cela nous allons utiliser un langage particulier, celui des règles d'inférence. Une règle d'inférence est une figure avec une barre de fraction:

$$\frac{P_1 \quad \ldots P_n}{C.}$$

Dans cette figure, les expressions qui apparaissent au numérateur sont des *prémisses*, l'expression qui apparait au dénominateur est la conclusion. La figure se lit simplement *si les prémisses sont vraies alors la conclusion est vraie.*

Pour les prémisses et la conclusion, nous utiliserons d'autres figures, que nous appellerons des jugements. Les jugements auront systématique la forme suivante:

$$\rho \vdash relation.$$

Dans un tel jugement, il y aura donc toujours un environnement, qui permet de fixer les valeurs des expressions qui apparaissent dans la relation.

Chaque jugement devra être lu comme une phrase de la langue naturelle. Voici les principaux jugements que nous allons rencontrer dans cette leçon:

- $\rho \vdash e \to v$, *l'expression* e *a la valeur* v,
- $\rho \vdash b \to v$, *l'expression* b *a la valeur* v,
- $\rho \vdash x, v \mapsto \rho'$, *la mise à jour de l'environnement associant la valeur* v *à la variable* x *retourne le nouvel environnement* ρ',
- $\rho \vdash i \rightsquigarrow \rho'$, *l'exécution de l'instruction* i *termine et retourne le nouvel environnement* ρ'.
- $\rho \vdash i \rightsquigarrow i', \rho'$ *l'exécution d'une étape élémentaire de calcul dans* i *laisse le programme* i' *à exécuter et associe les valeurs décrites dans* ρ' *aux variables du programme.*
- $\rho \vdash i \rightsquigarrow^* i', \rho'$ *l'exécution de plusieurs étapes élémentaires de calcul dans* i *laisse le programme* i' *à exécuter et associe les valeurs décrites dans* ρ' *aux variables du programme.*

Nous disposons maintenant de toutes les notations dont nous allons avoir besoin. Il reste à fournir les règles.

1.3 Evaluation des expressions

Les expressions les plus simples sont les valeurs entières, et leur valeur est elle-mÃªme (c'est toujours comme ça en informatique, on commence toujours par quelque chose de trivial).

$$\frac{}{\rho \vdash n \to n}.$$

La règle n'a pas de prémisse, mais on pourrait en mettre une: à condition que n soit vraiment un entier.

En augmentant un peu la complexité des expressions, on va trouver les variables. Pour ces variables, l'environnement (et sa structure) jouent un rôle:

$$\frac{}{(x, v) \cdot \rho \vdash x \to v},$$

$$\frac{x \neq y \vdash \rho \vdash x \to v}{(y, v') \cdot \rho \vdash x \to v}.$$

Ces règles indiquent nous allons utiliser l'environnement comme une liste ordonnée. On pourrait se donner d'autres présentations, comme la suivante:

$$\frac{(x, v) \in \rho}{\rho \vdash x \to v}$$

Une telle présentation impose que l'on travaille avec la convention que la mÃªme variable n'apparait pas plusieurs fois dans l'environnement, sinon la valeur associée à chaque variable ne serait pas unique.

Une troisième présentation consiste à considérer que la structure de l'environnement importe peu, mais que ce qui compte est qu'il représente une fonction des variables vers les valeurs. On pourrait alors écrire la règle suivante:

$$\frac{}{\rho \vdash x \to \rho(x)}.$$

La dernière règle pour les expressions arithmétiques concerne les additions:

$$\frac{\rho \vdash e_1 \to v_1 \quad \rho \vdash e_2 \to v_2}{\rho \vdash e_1 + e_2 \to v_1 + v_2}.$$

Pour cette règle nous n'avons pratiquement rien à dire, il faut toutefois noter la différence entre la construction syntaxique + qui prend deux expressions en paramètres et construit une nouvelle expression (sans calcul) et l'opération d'addition des entiers naturels, notée presque avec le même symbole.

Après l'évaluation pour les expressions arithmétiques, nous devons décrire l'évaluation pour les expressions booléennes. Deux approches sont possibles. La première approche consiste à considérer que l'on dispose de deux prédicats à deux arguments "$<$" et "$\leq$" dont la signification est intuitive et que l'on peut utiliser pour les prémisses d'une règle d'inférence. On peut alors exprimer la sémantique de la comparaison avec les règles suivantes:

$$\frac{\rho \vdash e_1 \to v_1 \quad \rho \vdash e_2 \to v_2 \quad v_1 < v_2}{\rho \vdash e_1 < e_2 \to \texttt{true}}$$

$$\frac{\rho \vdash e_1 \rightarrow v_1 \quad \rho \vdash e_2 \rightarrow v_2 \quad v_2 \leq v_1}{\rho \vdash e_1 < e_2 \rightarrow \texttt{false}}$$

C'est cette approche que nous utiliserons dans la suite de notre travail.

Présentons quand même la second approche, il s'agit ici de se munir d'une fonction *test_lt* à deux arguments entiers et retournant une valeur booléenne, telle que $test\_lt(v_1, v_2)$ soit `true` si $v_1 < v_2$ et `false` sinon. On peut alors exprimer la sémantique de l'opérateur de comparaison à l'aide d'une seule règle:

$$\frac{\rho \vdash e_1 \rightarrow v_1 \quad \rho \vdash e_2 \rightarrow v_2}{\rho \vdash e_1 < e_2 \rightarrow test\_lt(v_1, v_2)}$$

Les habitudes de la programmation (en C, Java, Caml, ou ML) sont de noter la fonction *test_lt* par le symbole `<`, mais il est important de savoir qu'il existe une différence le *prédicat* $<$ et la fonction qui permet de savoir si ce prédicat est satisfait. En particulier, il existe des prédicats pour lesquels il n'existe pas de fonction décidant la validité.

1.4 Mise à jour des environnements

L'effet des affectations est de modifier la valeur associée à des variables dans l'environnement. Nous représentons ceci à l'aide du jugement

$$\rho \vdash x, v \mapsto \rho'.$$

Si nous voulons mettre à jour l'environnement pour une variable qui apparait en tête de l'environnement, nous pouvons nous contenter de décrire un nouvel environnement où la valeur associée est changée:

$$\frac{}{(x, v) \cdot \rho \vdash x, v' \mapsto (x, v') \cdot \rho}$$

Cette règle exprime en même temps que les valeurs des autres variables sont inchangées.

Si la variable à modifier n'apparait pas en tête de l'environnement, la modification ne peut avoir lieu que dans le reste de l'environnement, mais la valeur associée à la première variable de l'environnement ne doit pas être changée.

$$\frac{\rho \vdash x, v \mapsto \rho' \quad (x \neq y)}{(y, v') \cdot \rho \vdash x, v \mapsto (y, v') \cdot \rho'}$$

1.5 Exécution des instructions

Pour bien comprendre la sémantique des instructions, il faut se souvenir de la signification du jugement "$\rho \vdash i \rightsquigarrow \rho'$", qui stipule simultanément que l'exécution termine et quel sera l'état final.

L'évaluation de l'instruction `skip` termine toujours et retourne toujours l'environnement de départ.

$$\frac{}{\rho \vdash \texttt{skip} \rightsquigarrow \rho}$$

Si l'évaluation de l'expression e retourne la valeur v et si la mise à jour de l'environnement pour la variable x et la valeur v retourne l'environnement ρ', alors l'exécution de l'instruction $x{:=}e$ termine et retourne l'environnement ρ'.

$$\frac{\rho \vdash e \to v \quad \rho \vdash x, v \mapsto \rho'}{\rho \vdash x{:=}e \rightsquigarrow \rho'}$$

La règle pour la séquence exprime que l'on accumule les effets des deux règles mises en séquence.

$$\frac{\rho \vdash i_1 \rightsquigarrow \rho' \quad \rho' \vdash i_2 \rightsquigarrow \rho''}{\rho \vdash i_1;i_2 \rightsquigarrow \rho''}$$

L'une des règles pour la boucle indique que l'on exécute le corps de la boucle si l'expression booléenne s'évalue à `true`, puis que l'on continue en tentant à nouveau l'exécution de la boucle.

$$\frac{\rho \vdash b \to \texttt{true} \quad \rho \vdash i \rightsquigarrow \rho' \quad \rho' \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho''}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho''}$$

L'autre règle pour la boucle exprime que le calcul s'arrête si l'expression booléenne s'évalue à `false`.

$$\frac{\rho \vdash b \to \texttt{false}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho}$$

Dans la suite, les cinq règles que nous venons de donner seront appelées SN1, SN2, ..., SN5.

2 La sémantique structurelle opérationnelle

Nous allons maintenant décrire une autre approche pour définir le comportement des instructions de notre langage. Cette fois-ci nous considérerons l'exécution d'étapes élémentaires d'un programme et nous considérerons ensuite les successions d'étapes élémentaires.

Pour décrire l'exécution d'une étape élémentaire, nous utiliserons le jugement

$$\rho \vdash i \rightsquigarrow i', \rho'.$$

Dans ce jugement, l'instruction à exécuter est i. L'instruction i' est utilisée pour décrire le calcul qui reste à faire après l'exécution d'une étape élémentaire. L'environnement ρ' est utilisé pour décrire l'effet sur les variables de l'étape élémentaire qui vient d'être exécutée. Voici les cinq règles décrivant des étapes élémentaires:

La règle pour l'affectation exprime qu'une étape élémentaire suffit à exécuter cette instruction dans son intégralité: il ne reste rien à faire, ce qui est représenté par l'instruction vide `skip`.

$$\frac{\rho \vdash e \to v \quad \rho \vdash x, v \mapsto \rho'}{\rho \vdash x{:=}e \rightsquigarrow \texttt{skip}, \rho'}$$

Il y a deux règles d'exécution élémentaire pour la séquence, pour exprimer que l'on doit d'abord chercher l'étape élémentaire dans la première instruction et ne passer à la second instruction que s'il ne reste rien à exécuter dans la première:

$$\frac{}{\rho \vdash \texttt{skip};i_2 \rightsquigarrow i_2, \rho}$$

$$\frac{\rho \vdash i_1 \rightsquigarrow i_1', \rho'}{rho \vdash i_1;i_2 \rightsquigarrow i_1';i_2, \rho'}$$

Pour la boucle, les deux règles expriment que l'étape élementaire de calcul consiste en la résolution du test sur l'expression booléenne.

$$\frac{\rho \vdash b \rightarrow \texttt{true}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow i;\texttt{while}\ b\ \texttt{do}\ i, \rho}$$

$$\frac{\rho \vdash b \rightarrow \texttt{false}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \texttt{skip}, \rho}$$

Il est important de noter qu'il n'existe pas de règle d'exécution élémentaire pour l'instruction `skip`.

Pour exécuter plusieurs étapes élémentaires, nous utilisons le jugement

$$\rho \vdash i \rightsquigarrow^* i', \rho'.$$

Ce jugement représente l'exécution de 0, 1, ou plusieurs étapes élémentaires. Ce jugement est défini par les deux règles suivantes:

$$\frac{}{\rho \vdash i \rightsquigarrow^* i, \rho}$$

$$\frac{\rho \vdash i \rightsquigarrow i', \rho' \quad \rho' \vdash i' \rightsquigarrow^* i'', \rho''}{\rho \vdash i \rightsquigarrow^* i'', \rho''}$$

Dans la suite, nous appellerons SOS1, SOS2, ..., SOS7, les sept règles d'inférence introduites dans cette section.

3 Dérivations et preuves

Les règles d'inférence sont des figures élémentaires de raisonnement. Elles peuvent être combinées pour obtenir des raisonnements complexes. Voici un exemple, qui décrit le cas d'exécution d'une boucle où l'on sait que l'expression booléenne s'évalue à `true` au moins deux fois:

$$\frac{\rho \vdash b \rightarrow \texttt{true}\ \rho \vdash i \rightsquigarrow \rho' \quad \dfrac{\rho' \vdash b \rightarrow \texttt{true}\ \rho' \vdash i \rightsquigarrow \rho''\ \rho'' \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho'''}{\rho' \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho'''}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho'''.}$$

Une telle figure est appelée une *dérivation partielle*. Celle-ci est obtenue en utilisant deux fois la règle SN4. Elle est partielle parce que certaines prémisses ne sont pas couvertes par dérivations.

Certaines règles, en particulier les règles SN1 et SN5, n'ont pas de prémisses. Ces règles permettent donc de prouver certaines des prémisses d'une dérivation partielle. Lorsque l'on parlera d'une dérivation (sans préciser), on supposera systématiquement qu'elle n'est pas partielle. Les dérivations sont des preuves complètes de jugements. Les jugements prouvables sont ceux pour lesquels il existe une dérivation dont la conclusion exprime ces jugements.

Les dérivations sont des objets à part entière, on peut également raisonner sur ces dérivations, faire des calculs sur ces dérivations, définir des fonctions qui prennent des dérivations en argument et retournent des dérivations. En particulier, nous définissons la fonction de taille d'une dérivation, qui compte le nombre de règles qui apparaissent dans une dérivation. Nous utiliserons les variables D, D', δ, δ_1, dots pour parler des dérivations.

4 Démonstrations sur les dérivations

Nous avons donné deux définitions du langage IMP. Nous voulons maintenant démontrer que ces deux définitions équivalentes. En fait, nous allons démontrer que si le jugement

$$\rho \vdash i \rightsquigarrow \rho'.$$

est prouvable, alors le jugement

$$\rho \vdash i \rightsquigarrow^* \texttt{skip}, \rho'$$

l'est également, et réciproquement.

Il y a en fait deux preuves, puisque l'équivalence représente deux implications. En outre nous aurons besoin de plusieurs lemmes, mais les lemmes 3 et 5 sont les lemmes principaux.

Lemme 1. $\forall \rho, \rho', \rho'', i, i', i'', \rho \vdash i \rightsquigarrow^* i', \rho' \wedge \rho' \vdash i' \rightsquigarrow^* i'', \rho'' \Rightarrow \rho \vdash i \rightsquigarrow^* i'', \rho''$.

Démonstration. Il s'agit de raisonner sur les dérivations qui permettent d'établir les différents jugements apparaissant dans l'énoncé du lemme. Nous pouvons supposer qu'il existe deux dérivations pour les énoncés donnés en hypothèse et nous devons montrer qu'il existe une dérivation pour le jugement donné en conclusion du lemme. Nous allons raisonner par récurrence sur la taille de la première dérivation existante. Nous ré-exprimons l'énoncé de la façon suivante: "pour toute dérivation D prouvant un énoncé de la forme $\rho \vdash i \rightsquigarrow^* i', \rho'$, s'il existe une dérivation D' prouvant un énoncé de la forme $\rho \vdash i' \rightsquigarrow^* i'', \rho''$, alors il existe une dérivation prouvant un énoncé de la forme $\rho \vdash i \rightsquigarrow^* i'', \rho''$".

Nous allons démontrer cet énoncé par récurrence sur la taille de la dérivation D. Nous pourrons donc supposer que le lemme est déjà établi pour toute dérivation de taille strictement inférieure à D.

Observons maintenant la dérivation D. Elle peut être obtenue avec l'une des règles SOS6 et SOS7 (on n'observe ici que la règle appartenant en bas de la dérivation). Si la règle SOS6 a été utilisée, alors les instructions i et i' sont égales et les environnements ρ et ρ' sont égaux. La dérivation D' est alors aussi une dérivation prouvant un énoncé de la forme $\rho \vdash i \rightsquigarrow^* i'', \rho''$.

Si la règle SOS7 a été utilisée, alors il existe une instruction i_1 et un environnement ρ_1 tels que les jugements $\rho \vdash i \rightsquigarrow i_1, \rho_1$ et $\rho_1 \vdash i_1 \rightsquigarrow^* i', \rho'$ soient tous les deux prouvables. Apellons δ_1 et δ_2 les dérivations correspondantes. La dérivation δ_2 est une sous dérivation de la dérivation D, sa taille est strictement inférieure, et nous pouvons utiliser l'hypothèse de récurrence pour obtenir une dérivation δ_3 qui prouve l'énoncé $\rho_1 \vdash i_1 \rightsquigarrow i'', \rho''$. En utilisant les dérivations δ_1 δ_3 et la règle SOS7, nous pouvons

reconstruire une dérivation pour le jugement $\rho \vdash i \rightsquigarrow i'', \rho''$. Ceci conclut notre démonstration.

Lemme 2. $\forall \rho, \rho', i_1, i_2, i_3,$
$\rho \vdash i_1;\{i_2;i_3\} \rightsquigarrow^* \texttt{skip}, \rho' \Leftrightarrow \rho \vdash \{i_1;i_2\};i_3 \rightsquigarrow^* \texttt{skip}, \rho'$.

Démonstration. Nous ne démontrerons que le sens $\Rightarrow$ l'autre direction est similaire et laissée en exercice. Nous démontrons ceci par récurrence sur la taille de la dérivation D pour $\rho \vdash i_1;\{i_2;i_3\} \rightsquigarrow^* \texttt{skip}, \rho'$. L'hypothèse de récurrence est que pour toute dérivation de taille inférieure à D prouvant un énoncé de la forme $\rho_a \vdash i_a;\{i_b;i_c\} \rightsquigarrow^* \texttt{skip}, \rho_b$ il existe une dérivation prouvant $\rho_a \vdash \{i_a;i_b\};i_c \rightsquigarrow^* \texttt{skip}, \rho_b$.

Observons la dérivation D elle a nécessairement l'une de deux formes. La première forme possible est la suivante:

$$\dfrac{\dfrac{}{\rho \vdash \texttt{skip};\{i_2;i_3\} \rightsquigarrow i_2;i_3, \rho} \qquad \begin{matrix}\delta \\ \rho \vdash i_2;i_3 \rightsquigarrow^* \texttt{skip}, \rho'\end{matrix}}{\rho \vdash \texttt{skip};\{i_2;i_3\} \rightsquigarrow \texttt{skip}, \rho'}$$

Dans ce cas, $i_1 = \texttt{skip}$. On peut alors construire la dérivation suivante, qui est la solution pour ce cas:

$$\dfrac{\dfrac{\dfrac{}{\rho \vdash \texttt{skip};i_2 \rightsquigarrow i_2, \rho}}{\{\texttt{skip};i_2\};i_3 \rightsquigarrow i_2;i_3} \qquad \begin{matrix}\delta \\ \rho \vdash i_2;i_3 \rightsquigarrow^* \texttt{skip}, \rho'\end{matrix}}{\rho \vdash \{\texttt{skip};i_2\};i_3 \rightsquigarrow^* \texttt{skip}, \rho'}$$

La deuxième forme possible pour la dérivation D est la suivante:

$$\dfrac{\dfrac{\begin{matrix}\delta_1 \\ \rho \vdash i_1 \rightsquigarrow i_1', \rho_1\end{matrix}}{\rho \vdash i_1;\{i_2;i_3\} \rightsquigarrow i_1';\{i_2;i_3\}, \rho_1} \qquad \begin{matrix}\delta_2 \\ \rho_1 \vdash i_1';\{i_2;i_3\} \rightsquigarrow^* \texttt{skip}, \rho'\end{matrix}}{i_1;\{i_2;i_3\} \rightsquigarrow^* \texttt{skip}, \rho'}$$

Dans ce cas la dérivation δ_2 satisfait les conditions de l'hypothèse de récurrence et nous pouvons en déduire qu'il existe une dérivation δ_3 qui prouve le jugement $\rho \vdash \{i_1';i_2\};i_3 \rightsquigarrow^* \texttt{skip}, \rho'$. Nous pouvons alors construire la dérivation suivante:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\delta_1 \\ \rho \vdash i_1 \rightsquigarrow i_1', \rho_1\end{matrix}}{\rho \vdash i_1;i_2 \rightsquigarrow i_1';i_2, \rho_1}}{\rho \vdash \{i_1;i_2\};i_3 \rightsquigarrow \{i_1';i_2\};i_3, \rho_1} \qquad \begin{matrix}\delta_3 \\ \rho_1 \vdash \{i_1';i_2\};i_3 \rightsquigarrow^* \texttt{skip}, \rho'\end{matrix}}{\rho \vdash \{i_1;i_2\};i_3 \rightsquigarrow^* \texttt{skip}, \rho'}$$

Ceci termine le sens $\Rightarrow$ de la démonstration. Le sens $\Leftarrow$ est similaire et laissé en exercice.

Lemme 3. $\forall \rho, \rho', i, \rho \vdash i \rightsquigarrow \rho' \Rightarrow \forall i', \rho'', \rho' \vdash i' \rightsquigarrow^* \texttt{skip}, \rho'' \Rightarrow \rho \vdash i;i' \rightsquigarrow^* \rho''$.

Démonstration. Appelons D la dérivation qui permet d'obtenir $\rho \vdash i \rightsquigarrow \rho'$. Nous ferons cette démonstration par récurrence sur la taille de D. L'hypothèse de récurrence sera la suivante: "pour toute dérivation δ prouvant un jugement de la forme $\rho_1 \vdash i_1 \rightsquigarrow \rho_2$ de taille strictement inférieure à D, et pour toute dérivation prouvant un énoncé $\rho_2 \vdash i' \rightsquigarrow^* \texttt{skip}, \rho''$ il existe une dérivation prouvant $\rho_1 \vdash i_1;i' \rightsquigarrow^* \texttt{skip}, \rho''$.

Fixons donc une dérivation D', une instruction i' et un environnement ρ'' tels que D' soit une dérivation pour le jugement $\rho' \vdash i' \rightsquigarrow^* \texttt{skip}, \rho''$, nous devons montrer que nous pouvons construire une dérivation pour $\rho \vdash i;i' \rightsquigarrow^* \rho''$.

Examinons les cinq cas possible pour la règle qui conclut la dérivation D.

SN1. Si la règle SN1 a été utilisée, alors $i = \texttt{skip}$, $\rho = \rho'$ et l'on peut construire la dérivation suivante:

$$\dfrac{\dfrac{}{\rho \vdash \texttt{skip};i' \rightsquigarrow i', \rho} \qquad \begin{matrix} D' \\ \rho \vdash i' \rightsquigarrow^* \texttt{skip}, \rho'' \end{matrix}}{\rho \vdash \texttt{skip};i' \rightsquigarrow^* \texttt{skip}, \rho''}$$

SN2. Si la règle SN2 a été utilisée, alors $i = x{:=}e$ et il existe une valeur v et des dérivations δ_1 et δ_2 pour les jugements $\rho \vdash e \rightarrow v$ et $\rho \vdash x, v \mapsto \rho'$. En utilisant les dérivations δ_1 et δ_2 et les règles SOS1, SOS2, SOS3, et SOS7, nous pouvons alors reconstruire une dérivation qui a la forme suivante:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\delta_1 \\ \rho \vdash e \rightarrow v\end{matrix} \qquad \begin{matrix}\delta_2 \\ \rho \vdash x, v \mapsto \rho'\end{matrix}}{\rho \vdash x{:=}e \rightsquigarrow \texttt{skip}, \rho'}}{\rho \vdash x{:=}e;i' \rightsquigarrow \texttt{skip};i', \rho'} \qquad \dfrac{\dfrac{}{\rho' \vdash \texttt{skip};i' \rightsquigarrow i', \rho'} \qquad \begin{matrix} D' \\ \rho' \vdash i' \rightsquigarrow^* \texttt{skip}, \rho'' \end{matrix}}{\rho' \vdash \texttt{skip};i' \rightsquigarrow^* \texttt{skip}, \rho''}}{\rho \vdash x{:=}e;i' \rightsquigarrow^* \texttt{skip}, \rho''}$$

SN3 si la règle SN3 a été utilisée alors $i = i_1;i_2$ et il existe un environnement ρ_1 et deux dérivations δ_1 et δ_2 prouvant les énoncés $\rho \vdash i_1 \rightsquigarrow \rho_1$ et $\rho_1 \vdash i_2 \rightsquigarrow \rho_2$. Ces deux dérivations δ_1 et δ_2 sont de taille strictement inférieure à D et on peut utiliser des hypothèses de récurrence pour ces dérivation. On utilise d'abord l'hypothèse de récurrence pour la dérivation δ_2 en utilisant la dérivation D' comme dérivation supplémentaire on obtient alors une dérivation δ_3 qui prouve l'énoncé $\rho_1 \vdash i_2;i' \rightsquigarrow^* \texttt{skip}, \rho''$.

Nous pouvons maintenant réutiliser l'hypothèse de récurrence pour la dérivation δ_1 et la dérivation δ_3 comme dérivation supplémentaire. Nous obtenons alors une dérivation δ_4 qui prouve le jugement $\rho \vdash i_1;\{i_2;i'\} \rightsquigarrow^* \texttt{skip}, \rho''$. Le lemme 2 suffit alors pour conclure ce cas.

SN4. Dans ce cas, $i = \texttt{while}\ b\ \texttt{do}\ i_1$, il existe une dérivation δ_1 prouvant l'énoncé $\rho \vdash b \rightarrow \texttt{true}$, une valeur ρ_1 et deux dérivations δ_2 et δ_3 prouvant les énoncés $\rho \vdash i_1 \rightsquigarrow \rho_1$ et $\rho_1 \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \rho'$. Les dérivations δ_2 et δ_3 remplissent les conditions pour l'hypothèse de récurrence. En utilisant cette hypothèse pour δ_3 et D' on obtient une dérivation δ_4 qui prouve

$$\rho_1 \vdash \texttt{while}\ b\ \texttt{do}\ i_1;i' \rightsquigarrow^* \texttt{skip}, \rho''$$

En utilisant l'hypothèse de récurrence pour δ_2 et δ_4 on obtient une dérivation pour δ_5 qui prouve

$$\rho_1 \vdash i_1;\{\texttt{while}\ b\ \texttt{do}\ i_1;i'\} \rightsquigarrow^* \texttt{skip}, \rho''$$

En utilisant le lemme 2, on obtient une dérivation δ_6 qui prouve

$$\rho_1 \vdash \{i_1;\{\texttt{while}\ b\ \texttt{do}\ i_1\};i' \rightsquigarrow^* \texttt{skip}, \rho'' \equiv J$$

On peut alors construire la dérivation suivante qui suffit pour conclure ce cas:

$$\dfrac{\dfrac{\dfrac{\dfrac{\delta_1}{\rho \vdash b \rightarrow \texttt{true}}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i_1 \rightsquigarrow i_1;\texttt{while}\ b\ \texttt{do}\ i_1, \rho} \quad \delta_6}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i_1;i' \rightsquigarrow i_1;\texttt{while}\ b\ \texttt{do}\ i_1;i', \rho} \quad J}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i_1;i' \rightsquigarrow^* \texttt{skip}, \rho''}$$

SN5 Dans ce cas, $i = \texttt{while}\ b\ \texttt{do}\ i_1$, $\rho = \rho'$ et il existe une dérivation δ_1 qui prouve $\rho \vdash b \rightarrow \texttt{false}$. On peut alors construire la dérivation suivante:

$$\dfrac{\dfrac{\dfrac{\dfrac{\delta_1}{\rho \vdash b \rightarrow \texttt{false}}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i \rightsquigarrow \texttt{skip}, \rho}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i;i' \rightsquigarrow \texttt{skip};i', \rho} \quad \dfrac{\dfrac{}{\rho \vdash \texttt{skip};i' \rightsquigarrow i', \rho} \quad \dfrac{D'}{\rho \vdash i' \rightsquigarrow^* \texttt{skip}, \rho''}}{\rho \vdash \texttt{skip};i' \rightsquigarrow^* \texttt{skip}, \rho''}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i_1;i' \rightsquigarrow \rho''}$$

Ceci conclut le dernier cas de la preuve.

Lemme 4. $\forall \rho, \rho', i, \rho \vdash i;\texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho' \Leftrightarrow \rho \vdash i \rightsquigarrow^* \texttt{skip}, \rho'$.

Démonstration. Ici encore, nous ne démontrons que la partie $\Rightarrow$. Nous effectuons la démonstration par récurrence sur la taille de la dérivation démontrant $\rho \vdash i;\texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho'$.

Cette dérivation ne peut avoir que deux formes. Voici la première forme:

$$\dfrac{\dfrac{}{\rho \vdash \texttt{skip};\texttt{skip} \rightsquigarrow \texttt{skip}; \rho} \quad \dfrac{}{\rho \vdash \texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho}}{\rho \vdash \texttt{skip};\texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho}$$

Dans ce cas, $i = \texttt{skip}$, $\rho = \rho'$ et la règle SOS6 permet de conclure. La deuxième forme que peut prendre la dérivation est la suivante:

$$\dfrac{\dfrac{\dfrac{\delta_1}{\rho \vdash i \rightsquigarrow i_1, \rho_1}}{\rho \vdash i;\texttt{skip} \rightsquigarrow^* \rho \vdash i_1;\texttt{skip}, \rho_1} \quad \dfrac{\delta_2}{\rho_1 \vdash i_1;\texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho'}}{\rho \vdash i;\texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho'}$$

Dans ce cas δ_2 réunit les conditions pour appliquer l'hypothèse de récurrence et nous pouvons en déduire une dérivation δ_3 prouvant

$$\rho_1 \vdash i_1 \rightsquigarrow^* \texttt{skip}, \rho'$$

Nous pouvons alors construire la dérivation suivante, à l'aide de la règle SOS7:

$$\dfrac{\dfrac{\delta_1}{\rho \vdash i \rightsquigarrow i_1, \rho_1} \quad \dfrac{\delta_3}{\rho_1 \vdash i_1 \rightsquigarrow \texttt{skip}, \rho'}}{\rho \vdash i \rightsquigarrow^* \texttt{skip}, \rho'}$$

Le sens opposé se démontre de façon similaire.

Théorème 1. $\forall \rho, \rho', i, \rho \vdash i \rightsquigarrow \rho' \Rightarrow \rho \vdash i \rightsquigarrow^* \texttt{skip}, \rho'$.

Démonstration. Supposons $\rho \vdash i \rightsquigarrow \rho'$. Il est évident que $\rho' \vdash \texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho'$, d'après le lemme 3 on peut en déduire $\rho \vdash i;\texttt{skip} \rightsquigarrow^* \texttt{skip}, \rho'$ et le lemme 4 suffit pour conclure.

Lemme 5. $\forall \rho, (\rho', i, \rho \vdash i \rightsquigarrow i', \rho' \wedge \forall \rho'', \rho' \vdash i' \rightsquigarrow \rho'') \Rightarrow \rho \vdash i \rightsquigarrow \rho''$.

Démonstration. Cette démonstration par récurrence sur la taille de la dérivation prouvant $\rho \vdash i \rightsquigarrow i', \rho'$. Il y a 5 cas.

SOS1. Dans ce cas, $i = x{:=}e$, $i' = \texttt{skip}$ et il existe une valeur v et deux dérivations δ_1 et δ_2 prouvant les énoncés $\rho \vdash e \rightarrow v$ et $\rho \vdash x, v \mapsto \rho'$. Toute dérivation de la forme $\rho' \vdash \texttt{skip} \rightsquigarrow \rho''$ est nécessairement telle que $\rho'' = \rho'$ (cette dérivation est forcément basée sur la règle SN1) et l'on arrive donc à construire la dérivation suivante en utilisant la règle SN2, ce qui suffit à conclure ce cas:

$$\frac{\begin{array}{cc}\delta_1 & \delta_2 \\ \rho \vdash e \rightarrow v & \rho \vdash x, v \mapsto \rho'\end{array}}{\rho \vdash x{:=}e \rightsquigarrow \rho'}$$

SOS2. Dans ce cas, $i = \texttt{skip};i_2$, $i' = i_2$, $\rho = \rho'$. Si D' est une dérivation prouvant $\rho' \vdash i_2 \rightsquigarrow \rho''$, alors la dérivation suivante obtenue à l'aide des règles SN1 et SN3 suffit pour conclure ce cas:

$$\frac{\begin{array}{cc}\overline{\rho \vdash \texttt{skip} \rightsquigarrow \rho} & D' \\ & \rho \vdash i_2 \rightsquigarrow \rho''\end{array}}{\rho \vdash \texttt{skip};i_2 \rightsquigarrow \rho''}$$

SOS3. Dans ce cas $i = i_1;i_2$, $i' = i'_1;i_2$, et il existe une dérivation δ prouvant la dérivation $\rho \vdash i_1 \rightsquigarrow i'_1, \rho'$. Par ailleurs, la dérivation prouvant $\rho' \vdash i'_1;i_2 \rightsquigarrow \rho''$ est nécessairement obtenue à l'aide de la règle SN3 et il existe un environnement ρ_1 et deux dérivations δ_1 et δ_2 prouvant les énoncés $\rho' \vdash i'_1 \rightsquigarrow \rho_1$ et $\rho_1 \vdash i \rightsquigarrow \rho''$. La dérivation δ est de taille inférieure à la dérivation initiale et nous pouvons utiliser l'hypothèse de récurrence avec la dérivation δ_2. Il existe donc une dérivation δ_4 prouvant $\rho \vdash i_1 \rightsquigarrow \rho_1$. Nous pouvons donc construire la dérivation suivante à l'aide de la règle SN3. Cette dérivation suffit pour conclure ce cas:

$$\frac{\begin{array}{cc}\delta_4 & \delta_3 \\ \rho \vdash i_1 \rightsquigarrow \rho_1 & \rho \vdash i_2 \rightsquigarrow \rho''\end{array}}{\rho \vdash i_1;i_2 \rightsquigarrow \rho''}$$

SOS4. Dans ce cas $i = \texttt{while}\ b\ \texttt{do}\ i_1$, $i' = i_1;\texttt{while}\ b\ \texttt{do}\ i_1$ $\rho = \rho'$, et il existe une dérivation δ prouvant l'énoncé $\rho \vdash b \rightarrow \texttt{true}$. La dérivation prouvant $\rho \vdash i_1;\texttt{while}\ b\ \texttt{do}\ i_1 \rightsquigarrow \rho''$ a nécessairement été obtenue avec la règle SN3 et il existe donc un environnement ρ_1 et deux dérivations δ_1 et δ_2 prouvant les énoncés $\rho \vdash i_1 \rightsquigarrow \rho_1$ et $\rho_1 \vdash \texttt{while}\ b\ \texttt{do}\ i_1 \rightsquigarrow \rho''$. On peut alors construire la dérivation suivante en utilisant la règle SN4:

$$\frac{\begin{array}{ccc}\delta & \delta_1 & \delta_2 \\ \rho \vdash b \rightarrow \texttt{true} & \rho \vdash i_1 \rightsquigarrow \rho_1 & \rho_1 \vdash \texttt{while}\ b\ \texttt{do}\ i_1 \rightsquigarrow \rho''\end{array}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i_1 \rightsquigarrow \rho''}$$

SOS5. Dans ce cas $i = \texttt{while}\ b\ \texttt{do}\ i_1$, $i' = \texttt{skip}$, $\rho = \rho'$ et il existe une dérivation δ prouvant l'énoncé $\rho \vdash b \rightarrow \texttt{false}$. La dérivation $\rho \vdash \texttt{skip} \rightsquigarrow \rho''$ a nécessairement été

obtenue avec la règle SN1 et $\rho = \rho''$. On peut alors construire la dérivation suivante à l'aide de la règle SN5:

$$\frac{\begin{array}{c}\delta \\ \rho \vdash b \rightarrow \texttt{false}\end{array}}{\rho \vdash \texttt{while}\ b\ \texttt{do}\ i_1 \rightsquigarrow \rho}$$

Ceci termine le dernier cas de notre démonstration.

Théorème 2. $\forall \rho, \rho', i, \rho \vdash i \rightsquigarrow^* \texttt{skip}, \rho' \Rightarrow \rho \vdash i \rightsquigarrow \rho'$.

Démonstration. Par récurrence sur la taille de la dérivation pour $\rho \vdash i \rightsquigarrow^* \texttt{skip}, \rho'$. On peut observer cette dérivation, il y a deux cas:

SOS6. Si la dérivation a été obtenue par la règle SOS6, alors la $i = \texttt{skip}$ et $\rho = \rho'$, la règle SN1 suffit pour conclure.

SOS7. Si la dérivation a été obtenue par la règle SOS7, alors il existe une instruction i_1, un environnement ρ_1 et deux dérivations δ_1 et δ_2 prouvant les énoncés $\rho \vdash i \rightsquigarrow i_1, \rho_1$ et $\rho \vdash i_1 \rightsquigarrow^* \texttt{skip}, \rho'$. La dérivation δ_2 est de taille inférieure à la dérivation initiale et l'on peut utiliser l'hypothèse de récurrence pour obtenir une dérivation δ_3 prouvant l'énoncé $\rho_1 \vdash i_1 \rightsquigarrow \rho'$. Le lemme 5 appliqué aux dérivations δ_1 et δ_3 permet alors de conclure.

5 Conclusion

Les dérivations sont des "traces d'exécution symbolique" pour les descriptions sémantiques correspondantes. La preuve d'équivalence entre les deux sémantiques consiste à mettre en correspondance toutes les exécutions possibles avec l'une des sémantiques avec des exécutions possibles avec l'autre sémantique.

Il y a de nombreux cas, toujours autant qu'il y a de règles dans la description sémantique, mais les techniques utilisées sont toujours les mêmes. Il s'agit de retrouver les prémisses de la dérivation dans la sémantique de départ, d'utiliser éventuellement l'hypothèse de récurrence sur ces prémisses et de reconstruire une nouvelle dérivation valide pour la sémantique d'arrivée. Le procédé qui consiste à observer la forme possible d'une dérivation pour retrouver ses prémisses porte le nom d'*inversion*.

Ce type de démonstration se vérifie très bien par ordinateur. L'utilisation de la machine permet aussi d'automatiser certaines étapes.

References

[1] Hanne Riis Nielson, Flemming Nielson, *Semantics with application, A formal Introduction*, John Wiley & Sons, 1992, *accessible sur internet à l'URL* `http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html`

[2] Glynn Winskel, *Formal semantics of programming languages an introduction*, AAAI press, 1993.