

INF889J Semaine 1 (Solutionnaire)

Soit les expressions binaires *simples* générées par la grammaire (de syntaxe concrète) suivante :

$$b ::= 0 \mid 1 \mid b1 \mid b0 \quad (1)$$

Ces expressions représentent les nombres naturels en binaire. Par exemple, on peut encoder :

- le nombre 3 comme 11
- le nombre 4 comme 100
- le nombre 6 comme 110

On pourrait lui donner la syntaxe abstraite suivante :

$$b ::= 0 \mid 1 \mid \text{lshiftPlus1}(b) \mid \text{lshift}(b)$$

Cependant, pour faciliter la lecture, nous utiliserons la syntaxe concrète ci-dessous.

PROBLÈME 1. *Donnez une sémantique dénotationnelle $\llbracket \cdot \rrbracket$ qui interprète les expressions binaires simples en tant que nombres naturels. Vous pouvez utiliser au choix la syntaxe concrète ou abstraite. Elle devrait satisfaire, par exemple, les identités $\llbracket 11 \rrbracket = 3$ et $\llbracket 100 \rrbracket = 6$.*

Solution 0: Soit la sémantique suivante, où $\llbracket \cdot \rrbracket : [\text{Bin}] \rightarrow \mathbb{N}$ est une fonction totale qui interprète les expressions binaires simples en tant que nombres naturels.

$$\llbracket 0 \rrbracket = 0$$

$$\llbracket 1 \rrbracket = 1$$

$$\llbracket b0 \rrbracket = 2\llbracket b \rrbracket$$

$$\llbracket b1 \rrbracket = 2\llbracket b \rrbracket + 1$$

□

PROBLÈME 2. *Supposons qu'on ajoute un opérateur $\text{succ}(b)$ au langage, qui permet de calculer le successeur de b (la valeur qui correspond à « $b + 1$ ») :*

$$b ::= 0 \mid 1 \mid b1 \mid b0 \mid \text{succ}(b)$$

Donnez une sémantique opérationnelle à petite étapes qui permet de réduire toute expression à une valeur (c'est-à-dire, qui élimine les successeurs) :

(1) *Définissez un jugement qui spécifie quelles expressions sont des valeurs. Les valeurs de notre langage d'expression binaires sont les expressions binaires simples générées par la grammaire (1).*

(2) *Définissez ensuite les réductions nécessaires.*

Ajoutez une interprétation $\llbracket \text{succ}(b) \rrbracket$ à votre sémantique dénotationnelle de la question précédente. Elle devrait satisfaire, par exemple, l'identité $\llbracket \text{succ}(11) \rrbracket = 4$.

Solution 0: Soit les jugements $b \text{ val}$ et $b \mapsto b'$ définis par les règles suivantes :

$\boxed{b \text{ val}}$ b est une valeur

$$\frac{}{0 \text{ val}} \quad \frac{}{1 \text{ val}} \quad \frac{b \text{ val}}{b1 \text{ val}} \quad \frac{b \text{ val}}{b0 \text{ val}}$$

$\boxed{b \mapsto b'}$ b se réduit à b' en une étape

$$\begin{array}{c} \frac{b \mapsto b'}{b0 \mapsto b'0} (\mapsto 0) \quad \frac{b \mapsto b'}{b1 \mapsto b'1} (\mapsto 1) \quad \frac{b \mapsto b'}{\text{succ}(b) \mapsto \text{succ}(b')} (\mapsto \text{succ}) \\[10pt] \frac{}{\text{succ}(0) \mapsto 1} (\mapsto 0) \quad \frac{}{\text{succ}(1) \mapsto 10} (\mapsto 1) \\[10pt] \frac{[b \text{ val}]}{\text{succ}(b0) \mapsto b1} (\mapsto b0) \quad \frac{[b \text{ val}]}{\text{succ}(b1) \mapsto \text{succ}(b)0} (\mapsto b1) \end{array}$$

Les prémisses entre crochets sont facultatives et rendent la réduction déterministe. La nouvelle dénotation est $\llbracket \text{succ}(b) \rrbracket = \llbracket b \rrbracket + 1$. $\square$

PROBLÈME 3 (PETIT DÉFI). Supposons au lieu qu'on ajoute un opérateur $\text{plus}(b_1; b_2)$ au langage, qui représente la somme de b_1 et b_2 :

$$b ::= 0 \mid 1 \mid b1 \mid b0 \mid \text{plus}(b_1; b_2)$$

Définissez une sémantique opérationnelle à grosses étapes qui permet de réduire toute expression à une expression binaire simple (c'est-à-dire, qui permet d'éliminer l'opérateur plus). Ajoutez ensuite une interprétation $\llbracket \text{plus}(b_1; b_2) \rrbracket$ à votre sémantique dénotationnelle.

Solution 0: Soit le jugement $b \Downarrow b'$ qui évalue b à b' , défini par induction par les règles suivantes :

$\boxed{b \Downarrow b'}$ b évalue à b'

$$\begin{array}{c} \frac{}{0 \Downarrow 0} \quad \frac{}{1 \Downarrow 1} \quad \frac{b \Downarrow b'}{b0 \Downarrow b'0} \quad \frac{b \Downarrow b'}{b1 \Downarrow b'1} \\[10pt] \frac{}{\text{plus}(0; b) \Downarrow b} \\[10pt] \frac{}{\text{plus}(1; 0) \Downarrow 1} \quad \frac{}{\text{plus}(1; 1) \Downarrow 10} \quad \frac{b \Downarrow b'}{\text{plus}(1; b0) \Downarrow b'1} \quad \frac{\text{plus}(1; b) \Downarrow b' \quad \text{plus}(1; b1) \Downarrow b'0}{\text{plus}(1; b) \Downarrow b'} \\[10pt] \frac{}{\text{plus}(b1; 0) \Downarrow b1} \quad \frac{\text{plus}(1; b) \Downarrow b' \quad \text{plus}(b; b') \Downarrow b''}{\text{plus}(b1; 1) \Downarrow b'0} \quad \frac{\text{plus}(b; b') \Downarrow b''}{\text{plus}(b1; b'0) \Downarrow b''1} \quad \frac{\text{plus}(b; b') \Downarrow \beta \quad \text{plus}(\beta; 1) \Downarrow b''}{\text{plus}(b1; b'1) \Downarrow b''0} \\[10pt] \frac{}{\text{plus}(b0; 0) \Downarrow b0} \quad \frac{b \Downarrow b' \quad \text{plus}(b; b') \Downarrow b''}{\text{plus}(b0; 1) \Downarrow b'1} \quad \frac{\text{plus}(b; b') \Downarrow b''}{\text{plus}(b0; b'0) \Downarrow b''0} \quad \frac{\text{plus}(b; b') \Downarrow b''}{\text{plus}(b0; b'1) \Downarrow b''1} \end{array}$$

Ces règles implémentent l'algorithme d'addition du primaire, où l'on additionne les bits colonne par colonne de droite à gauche, et où l'on ajoute des retenues au besoin. $\square$

A Expressions

A.1 Grammaire

Où $n \in \mathbb{N}$, les arbres de syntaxe abstraite pour notre langage d'expressions sont donnés par la grammaire suivante :

$$\begin{aligned}
 e ::= & \text{num}[n] \\
 & | \text{true} \\
 & | \text{false} \\
 & | \text{plus}(e_1; e_2) \\
 & | \text{lt}(e_1; e_2) \\
 & | \text{if}(e_1; e_2; e_3)
 \end{aligned}$$

Soit $[\text{Exp}]$ l'ensemble des arbres de syntaxe abstraite générés par la grammaire ci-dessus.

A.2 Sémantique opérationnelle : petites étapes

$\boxed{e \text{ val}}$ e est une valeur

$$\frac{}{\text{num}[n] \text{ val}} \text{ (SOS-NUM)} \quad \frac{}{\text{true val}} \text{ (SOS-TRUE)} \quad \frac{}{\text{false val}} \text{ (SOS-FALSE)}$$

$\boxed{e_1 \mapsto e_2}$ e_1 se réduit à e_2 en une étape

$$\begin{aligned}
 & \frac{}{\text{plus}(\text{num}[m]; \text{num}[n]) \mapsto \text{num}[m+n]} \text{ (SOS-PLUS)} \\
 & \frac{e_1 \mapsto e'_1}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e'_1; e_2)} \text{ (SOS-PLUS-1)} \quad \frac{e_1 \text{ val} \quad e_2 \mapsto e'_2}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e_1; e'_2)} \text{ (SOS-PLUS-2)} \\
 & \frac{m < n}{\text{lt}(\text{num}[m]; \text{num}[n]) \mapsto \text{true}} \text{ (SOS-LT-TRUE)} \quad \frac{\neg(m < n)}{\text{lt}(\text{num}[m]; \text{num}[n]) \mapsto \text{false}} \text{ (SOS-LT-FALSE)} \\
 & \frac{e_1 \mapsto e'_1}{\text{lt}(e_1; e_2) \mapsto \text{lt}(e'_1; e_2)} \text{ (SOS-LT-1)} \quad \frac{e_1 \text{ val} \quad e_2 \mapsto e'_2}{\text{lt}(e_1; e_2) \mapsto \text{lt}(e_1; e'_2)} \text{ (SOS-LT-2)} \\
 & \frac{}{\text{if}(\text{true}; e_1; e_2) \mapsto e_1} \text{ (SOS-IF-TRUE)} \quad \frac{}{\text{if}(\text{false}; e_1; e_2) \mapsto e_2} \text{ (SOS-IF-FALSE)} \\
 & \frac{e_0 \mapsto e'_0}{\text{if}(e_0; e_1; e_2) \mapsto \text{if}(e'_0; e_1; e_2)} \text{ (SOS-IF)}
 \end{aligned}$$

A.3 Sémantique opérationnelle : grosses étapes

$$\boxed{e \Downarrow v}$$

e évalue à v

$$\begin{array}{c} \frac{}{\text{num}[n] \Downarrow \text{num}[n]} \text{ (EV-NUM)} \quad \frac{}{\text{true} \Downarrow \text{true}} \text{ (EV-TRUE)} \quad \frac{}{\text{false} \Downarrow \text{false}} \text{ (EV-FALSE)} \\[10pt] \frac{e_1 \Downarrow \text{num}[m] \quad e_2 \Downarrow \text{num}[n]}{\text{plus}(e_1; e_2) \Downarrow \text{num}[m + n]} \text{ (EV-PLUS)} \\[10pt] \frac{e_1 \Downarrow \text{num}[m] \quad e_2 \Downarrow \text{num}[n] \quad m < n}{\text{lt}(e_1; e_2) \Downarrow \text{true}} \text{ (EV-LT-TRUE)} \\[10pt] \frac{e_1 \Downarrow \text{num}[m] \quad e_2 \Downarrow \text{num}[n] \quad \neg(m < n)}{\text{lt}(e_1; e_2) \Downarrow \text{false}} \text{ (EV-LT-FALSE)} \\[10pt] \frac{e_1 \Downarrow \text{true} \quad e_2 \Downarrow v}{\text{if}(e_1; e_2; e_3) \Downarrow v} \text{ (EV-IF-TRUE)} \quad \frac{e_1 \Downarrow \text{false} \quad e_2 \Downarrow v}{\text{if}(e_1; e_2; e_3) \Downarrow v} \text{ (EV-IF-FALSE)} \end{array}$$

A.4 Sémantique dénotationnelle

La sémantique du langage est donnée par une fonction partielle $\llbracket \cdot \rrbracket : [\mathbf{Exp}] \rightarrow (\mathbb{N} \cup \mathbb{B})$, où $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$. On écrit $x \simeq y$ si :

- (1) x et y sont tous les deux définis et égaux ; ou
- (2) ni x , ni y sont définis

Par exemple, si $n \in \mathbb{N}$, alors $\llbracket e \rrbracket \simeq n$ signifie que $\llbracket e \rrbracket$ est définie et a la valeur n .

La fonction $\llbracket \cdot \rrbracket$ est définie par induction sur la structure des expressions :

$$\begin{aligned} \llbracket \text{num}[n] \rrbracket &= n \\ \llbracket \text{true} \rrbracket &= \mathbf{T} \\ \llbracket \text{false} \rrbracket &= \mathbf{F} \\ \llbracket \text{plus}(e_1; e_2) \rrbracket &= \llbracket e_1 \rrbracket + \llbracket e_2 \rrbracket \quad \text{s'il existe } m, n \in \mathbb{N} \text{ t.q. } \llbracket e_1 \rrbracket \simeq m \text{ et } \llbracket e_2 \rrbracket \simeq n \\ \llbracket \text{lt}(e_1; e_2) \rrbracket &= \begin{cases} \mathbf{T} & \text{s'il existe } m, n \in \mathbb{N} \text{ t.q. } \llbracket e_1 \rrbracket \simeq m \text{ et } \llbracket e_2 \rrbracket \simeq n \text{ et } m < n \\ \mathbf{F} & \text{s'il existe } m, n \in \mathbb{N} \text{ t.q. } \llbracket e_1 \rrbracket \simeq m \text{ et } \llbracket e_2 \rrbracket \simeq n \text{ et } m \geq n \end{cases} \\ \llbracket \text{if}(e_1; e_2; e_3) \rrbracket &= \begin{cases} \llbracket e_2 \rrbracket & \text{si } \llbracket e_1 \rrbracket \simeq \mathbf{T} \text{ et } \llbracket e_2 \rrbracket \text{ est définie} \\ \llbracket e_3 \rrbracket & \text{si } \llbracket e_1 \rrbracket \simeq \mathbf{F} \text{ et } \llbracket e_3 \rrbracket \text{ est définie} \end{cases} \end{aligned}$$

La dénotation n'est pas définie dans les autres cas.