Examen du cours "INTRODUCTION À LA COMPILATION"
Licence 3 – Université Paris Diderot Paris 7

Durée: 3 heures

Tout document non manuscrit autorisé.

Le soin apporté à la rédaction et à la présentation ainsi que la rigueur des réponses seront pris en compte dans la notation. Ce sujet se décompose en 2 parties indépendantes. La première est une application directe du cours d'analyse syntaxique. La seconde porte sur une extension du langage de programmation étudié en cours. On en rappelle la syntaxe abstraite :

e	$::=$		***Expression***
	$\mid$	$x, f, \ldots$	*Variable*
	$\mid$	$\mathbf{c}$	*Constante* $\mathbf{c} \in \mathbb{C}$
	$\mid$	$e\ e$	*Application*
	$\mid$	$\mathtt{fun}\ x \Rightarrow e$	*Fonction*
a, v	$::=$		***Valeurs***
	$\mid$	$x, f, \ldots$	*Variable*
	$\mid$	$\mathbf{c}$	*Constante* $\mathbf{c} \in \mathbb{C}$
	$\mid$	$\mathtt{fun}\ x \Rightarrow e$	*Fonction*

L'utilisation du sucre syntaxique [1] "`let` $x = e_1$ `in` e_2" est autorisée. On écrira aussi "`let` $f\ x_1\ \ldots\ x_n = e_1$ `in` e_2" pour l'expression "`let` $f =$ `fun` $x_1 \Rightarrow \ldots$ `fun` $x_n \Rightarrow e_1$ `in` e_2". L'ensemble des constantes $\mathbb{C}$ est formé des entiers, des opérations arithmétiques, des booléens, du branchement conditionnel et de l'opérateur de point fixe.

1 Analyse syntaxique (45min)

Soit la grammaire suivante :

S	$::=$	$A\ \#$
A	$::=$	$V\ =\ E$
	$\mid$	E
E	$::=$	V
V	$::=$	a
	$\mid$	$*\ E$

où :

- S, A, E et V sont des symboles non-terminaux ;
- a, $\#$, $=$ et $*$ sont des symboles terminaux.

[1] On rappelle qu'un sucre syntaxique est une construction syntaxique superficielle qui peut s'exprimer avec d'autres constructions (plus primitives) du langage. Les sucres syntaxiques sont généralement traduits directement dans les actions sémantiques des analyseurs syntaxiques.

Exercice 1

1. *Calculez les fonctions $FIRST$ et $FOLLOW$ associées à cette grammaire.*
2. *Donnez l'automate $LR(0)$ de cette grammaire.*
3. *Est-ce que cette grammaire est $LR(0)$? Justifiez votre réponse.*
4. *Construisez l'automate $LR(1)$ de cette grammaire.*

□

2 Enregistrements (2h15)

On étend la grammaire des termes du langage étudié en cours en rajoutant deux nouvelles constructions :

e	$::=$	$\ldots$	
	$\mid$	$\{\, r \,\}$	*Définition d'un enregistrement*
	$\mid$	$e.\ell$	*Extraction d'un champ d'enregistrement*
r	$::=$	ϵ	*Enregistrement vide*
	$\mid$	$\ell = e;\ r$	*Définition d'un champ*

La première construction permet de définir des **enregistrements**, similaires à ceux de CAML, à l'aide d'une liste de couples associant une expression e à un champ ℓ pris dans un ensemble $\mathcal{L}$ prédéfini. La seconde construction sert à **extraire** la valeur associée à un champ à l'intérieur d'un enregistrement. Le terme suivant est un exemple de programme valide utilisant ces mécanismes de manipulation des enregistrements :

$$\texttt{let}\ \mathit{move}\ dx\ dy\ p = \{x = p.x + dx; y = p.y + dy\}\texttt{in}$$
$$\mathit{move}\ 1\ 2\ \{x = 40 + 1; y = 40\}$$

Ici, on a choisi $\mathcal{L} \equiv \{x, y\}$.

Exercice 2 (Syntaxe concrète et syntaxe abstraite)

1. *Le terme "$\{\}$" est-il une expression valide ? Si oui, que représente-t-il ?*
2. *Le terme "$\{1 + 2\}$" est-il une expression valide ? Si oui, que représente-t-il ?*
3. *On se propose d'introduire une construction "$\{e$ `with` $r\}$" similaire à la construction* CAML *permettant de définir un nouvel enregistrement à partir de celui représenté par l'expression e dans lequel seuls certains champs sont redéfinis. Voici un exemple de session* CAML *utilisant ce mécanisme :*

```
# type person = { name : string; firstname : string; age : int };;
type person = { name : string; firstname : string; age : int; }
# let p1 = { name = "Skywalker"; firstname = "Luke"; age = 42 };;
val p1 : person = {name = "Skywalker"; firstname = "Luke"; age = 42}
# { p1 with name = "Vador"; age = 84 };;
- : person = {name = "Vador"; firstname = "Luke"; age = 84}
```

Est-ce que ce mécanisme peut être intégré à notre langage sous la forme d'un sucre syntaxique ? Pourquoi ?

4. *On veut maintenant autoriser la définition simultanée de plusieurs champs d'un enregistrement ayant la même valeur. Pour cela, on introduit la syntaxe concrète "$\ell_1, \ldots, \ell_n = e; r$". Si* CAML *proposait de telles expressions, voici un exemple de session qui serait autorisée :*

```
# let p3 = { name, firstname = "X"; age = 21 };;
val p3 : person = {name = "X"; firstname = "X"; age = 21 }
```

Cette construction est-elle un sucre syntaxique ? Pourquoi ?

□

Exercice 3 (Sémantique opérationnelle à grands pas) *On étend la syntaxe des valeurs en introduisant les enregistrements :*

$$
\begin{array}{rcll}
v & ::= & \ldots & \\
 & | & \{\, r \,\} & \textit{Enregistrement} \\
 & & & \\
r & ::= & \epsilon & \textit{Enregistrement vide} \\
 & | & \ell = v;\ r & \textit{Champ défini}
\end{array}
$$

1. *Donnez les règles de sémantique à grands pas (suivant une stratégie d'appel par valeur) du langage étudié en cours et donnez la signification du jugement d'évaluation qu'elles définissent.*
2. *Proposez une règle d'évaluation pour la définition d'un enregistrement. Vous pouvez décomposer cette règle en plusieurs sous-règles si vous le souhaitez.*
3. *Proposez une règle d'évaluation pour l'extraction d'un champ d'un enregistrement.*
4. *À l'aide de ces règles d'évaluation, construisez la dérivation d'évaluation du programme suivant :*

$$
\begin{array}{l}
\texttt{let}\ \mathit{swap}\ p = \\
\{h = p.l; l = p.h\} \\
\texttt{in} \\
\quad \mathit{swap}\ \{h = 1 + 1; l = 4\}
\end{array}
$$

5. *Quelles sont les expressions bloquées vis-à-vis de cette sémantique ?*

□

Exercice 4 (Compilation)

1. *Comment étendre la machine virtuelle vue en cours pour pouvoir y représenter des enregistrements ?*
2. *Proposez une extension du jeu d'instructions de la machine virtuelle pour y inclure les mécanismes de manipulation d'enregistrements.*
3. *Proposez une extension de la fonction de compilation vue en cours de façon à traiter les enregistrements.*

□

Exercice 5 (Sémantique statique) *On étend la syntaxe des types du λ-calcul simplement typé.*

$$
\begin{array}{rcll}
\tau & ::= & \ldots & \\
 & | & \{R\} & \textit{Type d'enregistrement} \\
 & & & \\
R & ::= & \epsilon & \textit{Type pour l'enregistrement vide} \\
 & | & \ell : \tau;\ R & \textit{Type pour un champ d'enregistrement}
\end{array}
$$

1. *Rappelez les règles de typage du langage étudié en cours et donnez la signification du jugement d'évaluation qu'elles définissent. À quoi sert le typage ?*
2. *Proposez une règle de typage pour la définition d'un enregistrement.*
3. *Proposez une règle de typage pour l'extraction d'un champ.*
4. *Pour avoir le droit de définir un enregistrement en* CAML, *il faut d'abord avoir déclaré son type en lui donnant un nom (dans notre exemple de session, il s'agissait du nom de type `person`). Pouvez-vous expliquer pourquoi ?*

□