Documentation de travail : Creation d'un compilateur pour le langage Léa

Michael Färber

13 décembre 2012

Résumé

Nous avons crée un compilateur pour le langage Léa, qui est capable de créer du code Java donné des fichiers Léa.

1 Cahier des charges précisé

Notre compilateur est capable de créer du code Java pour un fichier Léa donné. Un fichier Léa peut contenir des construits suivantes :

- Définition des constantes.
- Définitions des types.
- Définitions des fonctions.

Partout dans le code Léa, on peut écrire des commentaires de facon suivante :

```
// This is a one-line comment.

/* This is a more complex,
   tremendously long,
   not very expressive
   multi-line comment. */
```

Tout fichier Léa ne peut pas utiliser des constantes, types et fonctions définis dans les autres fichiers Léa.

1.1 Définition des constantes

On peut définir des constantes sans donner leur types, de facon suivante :

```
MEANING = 42;
PI = 3.14;
SHAKESPEARE = "To be or not to be ..."
FIBONACCI = [1, 2, 3, 5, 8];
TUPLE = (MEANING, PI, SHAKESPEARE, FIBONACCI);
```

1.2 Définitions des types

On peut définir des nouveaux types de facon suivante :

```
date = (int, int, int);

person = struct
{
        name : string;
        birthday : date;
};

person_list = struct
{
        elem : person;
        next : int_list;
};
```

Nous supportons l'utilisation des types suivants :

- bool : booléen
- int : entier
- float : virgule flottante
- string : chaîne de lettres
- list of TYPE : liste d'éléments de TYPE
- (TYPE_1, TYPE_2, ..., TYPE_N) : n-uplet
- struct { ELEM_1 : TYPE_1, ..., ELEM_N : TYPE_N } : structure

1.3 Définitions des fonctions

Nous distinguons deux cas différents : fonctions qui retournent une valeur (appelés procedure), et fonctions qui ne retournent pas une valeur (appelés function). Les fonctions peuvent prendre plusieurs arguments, qui sont rendu par valeur et pas par référence.

Nous ne permettons pas d'avoir des fonctions surchargés. Aussi, nous ne permettons pas des variables des types comme arguments pour les fonctions. Par contre, nous permettons des fonctions récurrentes.

```
procedure print_int_list(l : list of int)
{
        for i in [0 ... l.length()]
        {
                writeln(l[i].toString());
        }
}

function main(list of string) : int
{
        writeln("Hello World!");
```

```
        writeln("Fibonacci numbers:");
        print_int_list(FIBONACCI);

        return 0;
}
```

Tous les fichiers Léa doivent contenir une fonction main, qui prend une liste de chaînes de lettre et qui retourne un entier. Cette fonction est appelé au début du programme, avec des arguments donné.

Nous supportons des instructions suivants :

- VARIABLE := EXPRESSION;
- return EXPRESSION;
- FUNCTION(PAR_1, ..., PAR_N);
- if (BOOL_EXPRESSION) INSTRS
- if (BOOL_EXPRESSION) INSTRS ELSE INSTRS
- for VAR in [INT_EXPRESSION ... INT_EXPRESSION] INSTRS
- while (BOOL_EXPRESSION) INSTRS
- repeat INSTRS while (BOOL_EXPRESSION);