Théorie des langages de programmation
Le projet (II)

Alain Chillès – 祁冲

ParisTech Shanghai Jiao Tong
上海交大–巴黎高科卓越工程师学院

5 novembre 2020 – 2020年11月5日 –庚子九月二十

Plan

Le langage L_{AC}

Il est constitué

- une machine virtuelle, nommée VM, qui sera un tableau d'entiers ;
- une table des symboles, nommée LAC, qui sera un tableau d'entiers ;
- une pile de données, nommée data_stack, qui sera une pile LIFO d'entiers ;
- une pile de retour, nommée return_stack, qui sera une pile LIFO d'entiers ;

- tous ces objets seront gérés par un programme d'exécution, appelé run qui pourra fonctionner en mode interprété et en mode compilé.

Rappel sur les piles

On avait défini le type pile de la manière suivante :

Source C

```c
struct pile {
    int (*pop)(void);
    void (*push)(int);
};

typedef struct pile pile;
```

Donc, essentiellement deux possibilités pop (qui dépile) et push (qui empile).

Description du mode interprété

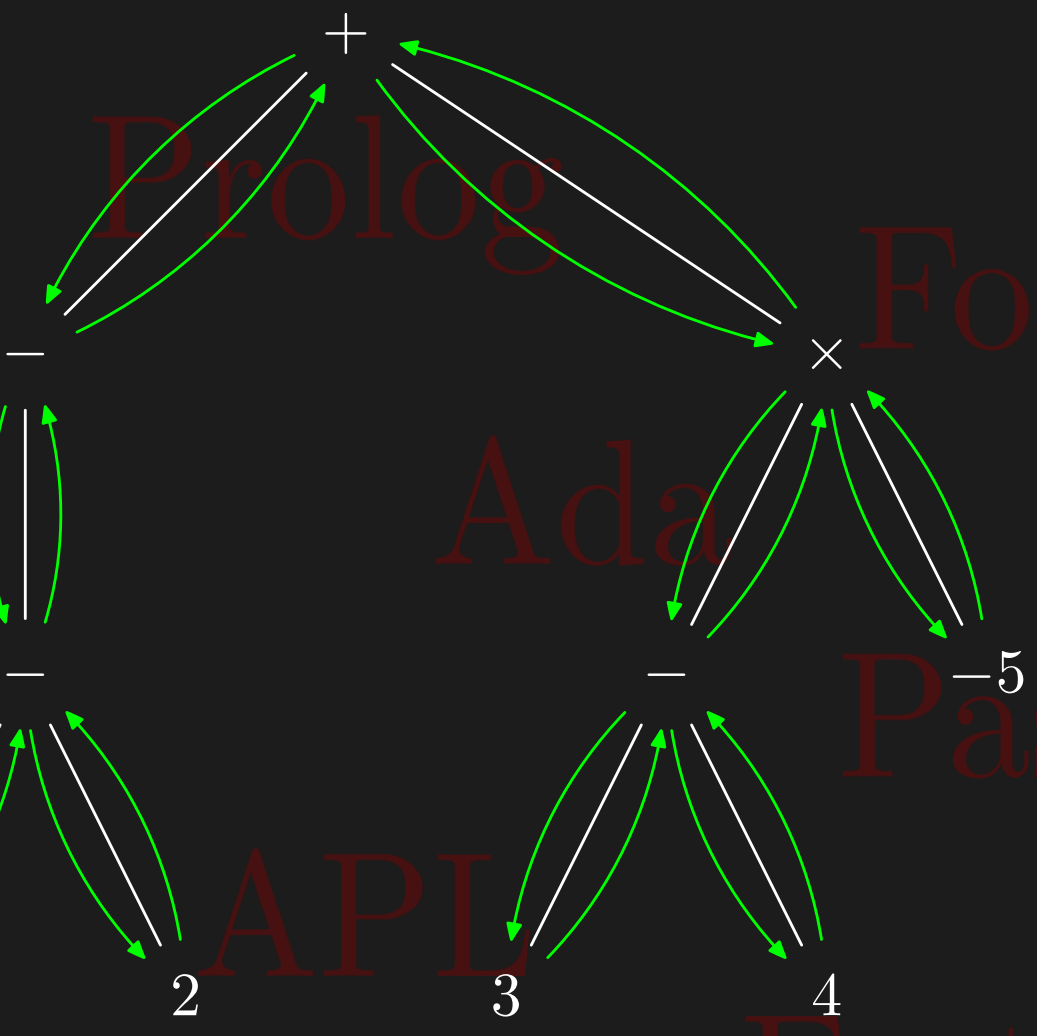
L$_{AC}$

```
1  " -(1-2)+(3-4)x(-5)" calculate .
```

La fonction `calculate` fait l'analyse lexicale, syntaxique et l'évaluation de l'expression, met sur la pile de données (`data_stack`) la valeur obtenue 6.

La fonction `.`, dépile la pile de données et imprime le résultat.

Rappels sur les arbres



Notations pré, post et infixées

Dans l'arbre, on passe en fait :

- une seule fois sur les feuilles ;
- deux fois sur les opérateurs unaires ;
- trois fois sur les opérateurs binaires.

Suivant le moment de lecture, l'arbre précédent peut se lire :

1. Notation infixée

   `-(1-2)+(3-4)x(-5)`

2. Notation préfixée

   `+␣-␣-␣1␣2␣x␣-␣3␣4␣-5` lu comme

   `(+ (- (- 1 2)) (x (- 3 4) -5))`

3. Notation postfixée

   `1␣2␣-␣-␣3␣4␣-␣-5␣x␣+` lu comme

   `(((1 2 -) -) ((3 4 -) -5 x) +)`

Plan

Deuxième étape

- Programmer la fonction `calculate` ;
- l'intégrer avec l'analyseur lexical de la semaine précédente, pour pouvoir exécuter la commande L_{AC}

L_{AC} – Calculatrice

```
" -(1-2)+(3-4)x(-5)" calculate .
```

Projet 2 : définition et contraintes

- Il s'agit ici de programmer une calculatrice qui pourra fonctionner en interprété (entiers signés, opérateurs $+$, $-$ et $\times$).
- Les fichiers à fournir sont :
  - Un fichier calculate.h
  - Un fichier calculate.c
  - Un fichier projet2.c qui permet de tester le code de `calculate`
- L'analyse lexicale se fera avec un automate
- On fournira dans le rapport
  - Une grammaire BNF, décrivant l'analyse syntaxique à faire
  - Un automate à pile, décrivant l'analyse syntaxique à faire
- On pourra utiliser l'une des deux méthodes au choix pour programmer l'analyseur syntaxique. Il devra fournir un arbre syntaxique.