Evaluation

Alexis Nasr
Franck Dary
Pacôme Perrotin

Compilation – L3 Informatique
Département Informatique et Interactions
Aix Marseille Université

Comment évaluer le compilateur ? I

- Méthode 1 : On programme le compilateur puis on compile des fichiers de test et on compare le résultat de leur exécution avec le résultat attendu.
- Inconvénients :
  - S'il y a une erreur, il est difficile de savoir d'où elle provient
  - On ne peut pas évaluer le compilateur avant de l'avoir terminé

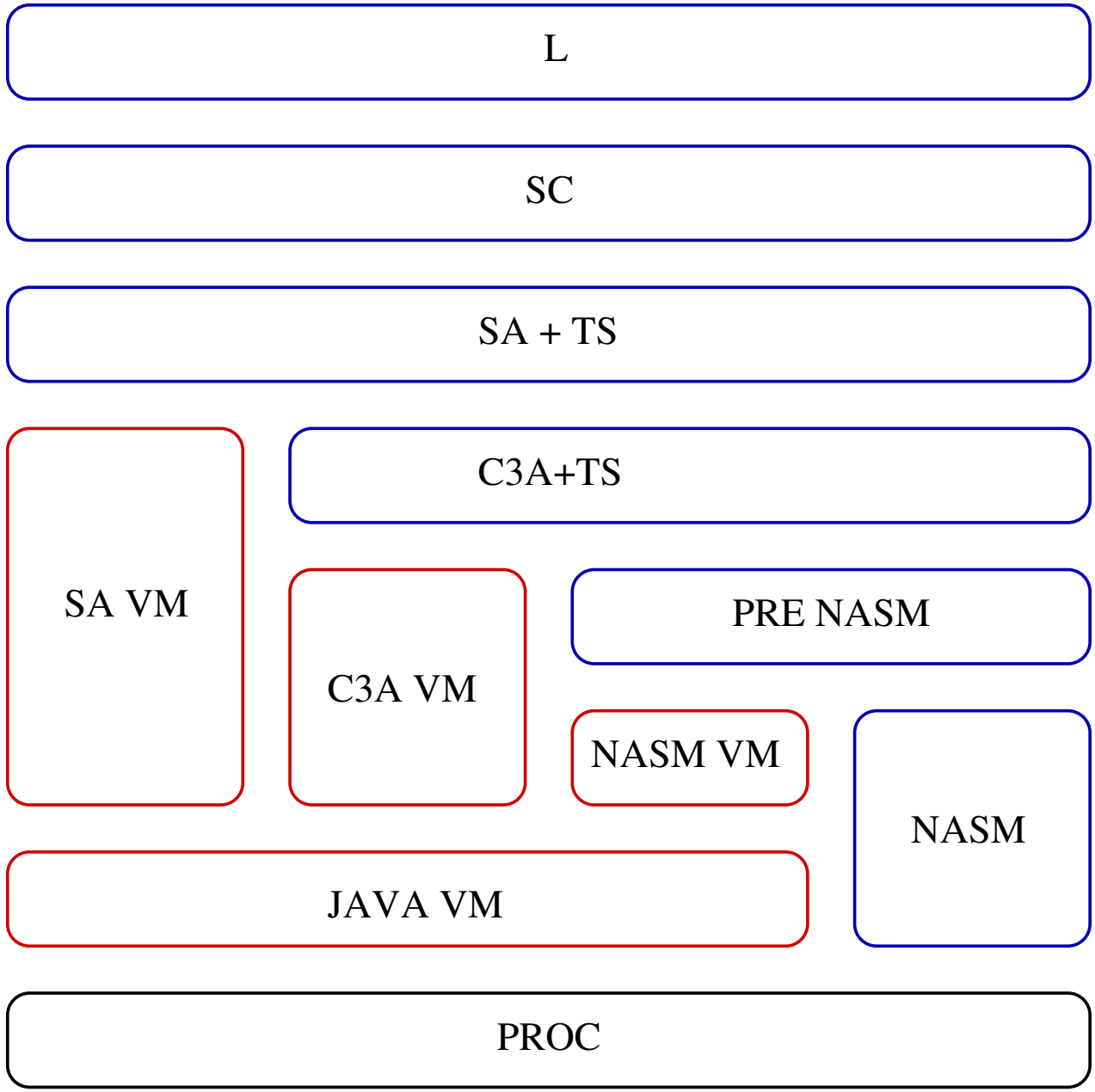
Comment évaluer le compilateur ? II

- Méthode 2 : On évalue les représentations des niveaux intermédiaires en les comparant avec les représentations attendues.
  - arbre de dérivation
  - arbre abstrait
  - table des symboles
  - code trois adresses
  - pré assembleur
  - assembleur
- Inconvénients :
  - Dans certains cas, les résultats intermédiaires obtenus ne sont pas exactement les mêmes que ceux de la référence
    - les grammaires peuvent être un peu différentes
    - les noms des temporaires dans le code trois adresses peuvent être différents
    - la traduction de certaines strucutres peuvent être différents
- Tant que la sémantique est la bonne, cela ne doit pas être un problème.

Comment évaluer le compilateur ? III

- Methode 3 : On exécute/interprète les représentations intermédiaires
  - syntaxe abstraite : `SaVM`
  - code trois adresses : `C3aVM`
  - pré-assembleur : `NasmVM`
  - assembleur : `NasmVM`
- C'est la méthode qu'on choisit ici.
- Pour la mettre en œuvre, on stocke sous la forme de fichiers textuels les différents niveaux de représentation :
  - syntaxe abstraite : arbre xml, extension des fichiers : `sa`
  - code trois adresses : format 'maison', extension des fichiers : `c3a`
  - pré-assembleur : syntaxe `Nasm`, au nom des registres près, extension des fichiers : `pre-nasm`
  - pré-assembleur : syntaxe `Nasm`, extension des fichiers : `nasm`

Rappel



Production des représentations intermédiaires

- On utilise l'option `-v 3` du compilateur.
- Pour chaque niveau de représentation intermédiaire, il existe une classe ou une méthode pour produire un fichier :
  - `public Sa2Xml(SaNode root, String baseFileName)`
  - `public void affiche(String baseFileName)` dans la classe `C3a`
  - `public void afficheNasm(String baseFileName)` dans la classe `Nasm`
- Exemple :

```
$java Compiler add1.l -v 2
[BUILD SC]
[PRINT SC]
[BUILD SA]
[PRINT SA]
[BUILD TS]
[PRINT TS]
[BUILD C3A]
[PRINT C3A]
[BUILD PRE NASM]
[PRINT PRE NASM]
[ALLOCATE REGISTERS]
[PRINT NASM]
```

Exécution des représentations intermédiaires

- Les trois machines virtuelles (`SaVM`, `C3aVM`, `NasmVM`) fonctionnent sur le même principe :
  1. Lecture du fichier correspondant au niveau intermédiaire (`.sa`, `.c3a`, `.pre-nams`, `.nasm`)
  2. Construction de la représentation en mémoire.
  3. Evaluation de la représentation en mémoire
     - `public SaEval(SaNode root, Ts tableGlobale)`
     - `public C3aEval(C3a c3a, Ts tableGlobale, int stackSize, int verboseLevel)`
     - `public NasmEval(Nasm code, int stackSize, int verboseLevel)`
  4. Ecriture du résultat de l'évaluation sur la sortie standard

Exécution des représentations intermédiaires : exemple

- Syntaxe abstraite

```
$java -cp ".:../xerces-2_12_1/*" SaVM -sa add1.sa
13
```

- Code trois adresses

```
$java C3aVM -c3a add1.c3a -ts add1.ts
13
```

- Pré-assembleur

```
$java NasmVM -nasm add1.pre-nasm
13
```

- Assembleur

```
$java NasmVM -nasm add1.nasm
13
```

Le répertoire `test`

Huit sous répertoires (entre autre) :

- `input` Contient 89 fichiers de test : des petits programmes en L
- `out-ref` Le résultat de l'exécution des programmes en L
- `sc-ref` Les arbres de dérivations des mêmes programmes
- `sa-ref` Les arbres abstraits
- `ts-ref` Les tables des symboles
- `c3a-ref` Le code trois adresses
- `pre-nasm-ref` Le code pré-assembleur
- `nasm-ref` Le code assembleur

Le script `evaluate.py`

Automatise le processus d'évaluation

1. Compile le compilateur et les trois machines virtuelles
2. Compile les programmes en L se trouvant dans `input`. Les résultats sont écrits dans `input`.
3. Compare les représentations intermédiaires produites avec les représentations de référence (celles qui se trouvent dans `*-ref`)

Ce script **doit** être utilisé pour l'évaluation, c'est lui qui sera utilisé pour l'évaluation finale du projet.

Exemple

```
$ python ./evaluate.py -s
SA-DIFF TS-DIFF SA     C3A-DIFF C3A    PRE-NASM-DIFF PRE-NASM NASM-DIFF NASM   EXE
100.00  100.00  100.00 100.00   100.00 100.00         98.82   100.00    100.00  98.82
------------------------------EVALUATION ERRORS------------------------------
ERROR evaluateSa for input varloc5.l : 'ATTENTION : la variable locale ou le paramètre var1 masque une variable globale
'
Sauvegardé dans le fichier result.txt
```

- Les chiffres correspondants aux colonnes `DIFF` ne servent pas à l'évaluation, ils sont là pour vous aider à comprendre ce qui se passe.
- Les colonnes importantes sont : `SA`, `C3A`, `PRE-NASM`, `NASM` et `EXE`

Les options du script `evaluate.py`

```
$ python ./evaluate.py -h
usage: evaluate.py [-h] [--verbose] [--silent] [--noColors] [--clean]
                   [--number NUMBER] [--files [FILES [FILES ...]]]

optional arguments:
  -h, --help            show this help message and exit
  --verbose, -v         Verbose output (obsolete, verbose is default).
  --silent, -s          Less verbose output.
  --noColors            Disable colors in output.
  --clean, -c           Clean input dir then exit.
  --number NUMBER, -n NUMBER
                        Restrict tests to n inputs.
  --files [FILES [FILES ...]], -f [FILES [FILES ...]]
                        Specify input files.
```

L'option `-f`

```
$ python ./evaluate.py -f add1.l
Compiling Compiler.java...Done
Compiling SaVM.java...Done
Compiling C3aVM.java...Done
Compiling NasmVM.java...Done
Compiling add1.l...Done
Évaluation de Diff de sa :
1/1 correct (100.00%)
add1.sa
Évaluation de Diff de ts :
1/1 correct (100.00%)
add1.ts
Évaluation de Execution de sa :
1/1 correct (100.00%)
add1.saout
Évaluation de Diff de c3a :
1/1 correct (100.00%)
add1.c3a
Évaluation de Execution du c3a :
1/1 correct (100.00%)
add1.c3aout
Évaluation de Diff de pre-nasm :
1/1 correct (100.00%)
add1.pre-nasm
Évaluation de Execution du pre-nasm :
1/1 correct (100.00%)
add1.pre-nasmout
Évaluation de Diff de nasm :
1/1 correct (100.00%)
add1.nasm
Évaluation de Execution du nasm :
1/1 correct (100.00%)
add1.nasmout
Évaluation de Execution du binaire :
1/1 correct (100.00%)
add1.exeout
Sauvegardé dans le fichier result.txt
```