
Introduction aux machines virtuelles

Cours 5 : Java Virtual Machine

Pierre Letouzey

Prenom.Nom@pps.jussieu.fr

Spécification de la JVM

La spécification de la JVM en ligne:

<http://java.sun.com/docs/books/jvms/>

Attention ! ne décrit que le comportement attendu de la JVM (sa *sémantique*). Les codages internes peuvent changer d'une implantation de JVM à une autre.

Traits principaux de la JVM

- **Portabilité:** plus de choses spécifiées qu'en Caml, p.ex. taille des types numériques. Même .class sur \neq archis.
Spec \pm ouverte: plusieurs JVM indépendantes.
Limitations: threads, graphisme
- **Mobilité et Sécurité:** typiquement, pouvoir exécuter des .class venant du web sans (trop de) risque(s).
Pour cela, vérificateurs de bytecode et bacs-à-sable, signatures éventuelles (\neq .cmo et ses appels système/C)

Aspects techniques

- **Typage**: pas totalement statique: object cast, instanceof.
Donc des types stockés dans les représentations (\neq Caml).
- **Valeurs**: variées (\neq Caml: “tout est un entier machine”)
Des myriades d'instructions selon les types:
istore, lstore, fstore, dstore, astore
Attention aux tailles dans la pile: p.ex. dup vs. dup2.
- **Tas**: comme en Caml, les données structurées sont allouées dans un tas (heap), et régies par un ramasse-miette (GC = Garbage Collector).
- **J.I.T.** (compilation Just-In-Time): certaines JVM compilent à la volée vers du code natif.

Organisation de la JVM

- Pour chaque thread, un pc et une pile, faite d'une suite de blocs (ou frame) pour les différentes fonctions en cours d'appel dans cette thread.
- Chaque frame a une taille fixe, connue à l'avance (\neq Caml)
 - une zone de variables locales. Au debut: args de la fn
 - une “operand stack”, de taille bornée à l'avance.
 - autres éléments, spécifiés (p.ex. renvoi à une zone de constantes) ou non spécifiés (sauvegarde pc de retour)
- De manière globale: une zone de tas partagée pour toutes les threads.

Suite de la comparaison JVM / Caml

- Des appels de fonctions à la fois plus simples (pas d'applications partielles) et plus complexes: selon le cas, `invokestatic`, `invokevirtual`, `invokespecial` ou `invokeinterface`.
- Appels récursifs possibles mais coûteux (pas de récursion terminal comme en Caml et .Net).
- La représentation des données doit signaler au GC ce qui est un pointeur ou non.
- Un `tableswitch/lookupswitch` similaire au `switch` Caml.
- prévu dans la spec de la VM: de la synchronisation (`monitor`)

Survol des instructions JVM

Voir la spécification de la JVM:

- section 3.11 pour une vision globale des instructions
- section 6 pour le descriptif précis de chaque instruction

Analyse d'un .class

Que contient Add.class ?

- En temps normal, un **déssassembleur** est suffisant:
`javap -c -verbose Add`
- Pour une fois, regardons le .class avec hexedit

Jasmin: un assembleur pour la JVM

Description de `Add2.j`, sa syntaxe, sa compilation via

```
jasmin Add2.j
```

Exemple de contrainte lié au vérification de `.class`:

la taille de pile