Université Paris 7 - L3 Informatique Année 2008–2009

Introduction aux machines virtuelles

Examen du 19 mai 2009

Durée : 3h. Documents autorisés

I) Caml (∼8 points)

Exercice 1 Traduire en bytecode Caml les phrases suivantes :

- `10-8*4/(3+1)`
- `let carre x = x*x in carre 5`
- `let a = 10 in let decuple x = a * x in decuple 5`
- `let f n = if n = 0 then 0 else n + f (n-1) in f 3`
- `let t = [|5;8|] in t.(0) + t.(1)`
  Donner deux manières d'accéder aux cases du tableau `t`. Laquelle de ces deux méthodes reste utilisable si on accède à la case `n` pour `n` non connu à l'avance ?

Exercice 2 Étant donné la portion de bytecode Caml suivant :

```
        closurerec 1, 0
        const [1: [1: [0: 10] [0: 20]] [0: 30]]
        push
        acc 1
        apply 1
        return 2
L1:     acc 0
        switch/ 3 2
L3:     acc 0
        getfield 0
        return 1
L2:     acc 0
        getfield 1
        push
        offsetclosure 0
        apply 1
        push
        acc 1
        getfield 0
        push
        offsetclosure 0
        apply 1
        addint
        return 1
```

On rappelle que la syntaxe `[n: a b ...]` correspond à la création d'un bloc de tag `n` et de données `a`, `b`, etc.

- Proposer un type caml correspondant à la donnée en cours de manipulation. Retrouver une phrase Caml dont la compilation produit ce bytecode. Associer à chaque élément de cette phrase la partie bytecode qui correspond. Quel est le résultat de l'exécution ?
- On exécute ce bytecode jusqu'au premier passage par l'instruction `offsetclosure`. Décrire alors l'état de la machine virtuelle Caml après cette instruction : contenu de l'accumulateur, de la pile et du tas. Justifier brièvement.
- Par quelle instruction équivalente mais plus efficace peut-on remplacer les instructions 5 et 6 (`apply 1` et `return 2`) ? Quel est l'effet de ce changement sur l'état de la pile demandé à la question précédente ?

II) Java (∼8 points)

Exercice 3 Traduire en bytecode Java les instructions suivantes (en sachant que `x` et `y` sont les variables locales d'indice 0 et 1).

- `y=10-8*4/(3+x);`
- `y=(3+x)*4/8-10;`
- `y=1; for (x=1; x<3; x++) y=y*x;`
- `int [] y = new int [2]; x = y[0]+y[1];`

Dans chacun des cas, simuler l'évolution de la pile lors de l'exécution (on suppose à chaque fois que `x` est nul initialement) et donner la taille de pile nécessaire pour le bon déroulement du calcul.

Exercice 4 Étant donné la portion de bytecode Java qui suit :

- Proposer une portion de code Java dont la compilation produit ce bytecode. Associer à chaque élément de code Java la partie bytecode qui correspond.
- Quel genre de données en entrée peut mener ce programme à une erreur. Ajouter des tests permettant d'empêcher cela.
- Proposer une variante équivalente au bytecode précédent, mais où le `goto` se situe entre deux labels et non avant.

```
.method public static histo([I)[I
      .limit locals 4
      .limit stack 4
      bipush 21
      newarray int
      astore_1
      iconst_0
      istore_2
      goto L2
L1:
      aload_0
      iload_2
      iaload
      istore_3
      aload_1
      iload_3
      aload_1
      iload_3
      iaload
      iconst_1
      iadd
      iastore
      iinc 2 1
L2:
      iload_2
      aload_0
      arraylength
      if_icmplt L1
      aload_1
      areturn
.end method
```

III) Problème : validation de bytecode ($\sim$ 6 points)

Par défaut, une JVM commence par vérifier que le bytecode Java qu'elle doit exécuter satisfait bien un certain nombre de critères de validité. En particulier, elle effectue un contrôle du nombre de variables locales et de cases de pile nécessaire à l'exécution du code. On cherche ici à voir comment ce contrôle peut être réalisé.

On suppose que le bytecode Java nous est donné sous forme d'un type abstrait Caml :

```
type instruction =
 | Iconst of int
 | Istore of int
 | Iload of int
 | Iadd
 | ...

type bytecode = instruction list
```

Exercice 5 Ecrire une fonction déterminant l'indice maximal des variables locales utilisées dans une portion de bytecode. Peut-il arriver que l'exécution de ce bytecode utilise en pratique moins de variable locales que cela ? Plus ?

Exercice 6 Quels problèmes se posent lorsque l'on tente de déterminer la quantité de pile maximale nécessaire pour exécuter une portion de bytecode ? Ce problème est-il soluble en toute généralité ? Décrire comment calculer cette taille maximale pour le cas des instructions n'entraînant pas de sauts.

Au lieu des véritables instructions de saut du bytecode Java, on suppose maintenant disposer d'instructions de plus haut niveau :

```
type instruction =
  ...
  | If_icmplt of bytecode * bytecode   (* code du then / du else *)
  | While of bytecode * bytecode * bytecode
            (* code d'initialisation / du test de fin / du corps de la boucle *)
  ...

and bytecode = instruction list
```

Exercice 7 Dans ce cadre, quelle contrainte doit satisfaire le bytecode se trouvant dans un `if` ou un `while` pour que l'on soit en mesure de déterminer une taille de pile maximale à l'avance ? Ecrire cette fonction de calcul de la taille de pile maximale.

Exercice 8 On souhaite également vérifier que chaque instruction trouvera bien une quantité de pile suffisante devant elle lors de l'exécution. Ainsi un `iadd` nécessite l'accès à deux cases de pile. Que faut-il changer aux questions précédentes pour être en mesure de tester cette propriété ?

Exercice 9 Ces techniques d'analyse de taille de piles peuvent-elles s'appliquer également dans le cas du bytecode Caml ?