

Machines virtuelles

Cours 1 : Introduction

Pierre Letouzey¹
`pierre.letouzey@inria.fr`

PPS - Université Denis Diderot – Paris 7

janvier 2012

1. Merci à Y. Régis-Gianas pour les transparents

Qu'est-ce qu'une machine virtuelle ?

L'art de la programmation

- | Informellement, la programmation est :

L'art de résoudre des problèmes efficacement, par le calcul.

- | Il existe de nombreux **modèles de calcul** que l'on peut par exemple classer selon les critères suivants :

<i>Critères</i>	<i>Exemples</i>
Digital Analogique	Ordinateurs standards ou quantiques Réseaux de neurones, mélanges chimiques
Séquentiel Parallèle	Programme C Programme Erlang
Automatique Supervisé	Calcul numérique Preuve assistée par ordinateur
Fonctionnel Impératif Logique Concurrent	λ -calcul Programme C Programme Prolog Système biologique

Une machine abstraite pour chaque modèle de calcul

- | Une machine abstraite décrit l'environnement d'évaluation d'un modèle de calcul.
- | Très peu de modèles possèdent une réalisation physique de leur machine abstraite (à l'exception du modèle de Von Neumann).
- | On peut cependant **émuler** une machine abstraite à l'aide d'un autre programme.
- | Un tel programme est appelé **machine virtuelle**.

Les machines virtuelles : une solution miracle ?

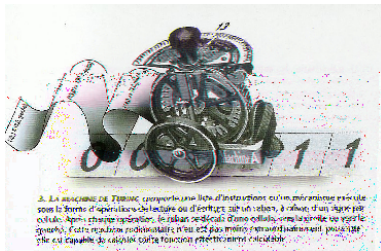
- | Une machine sans **existence physique** : ni silicium, ni engrenage.
 - | **Dématérialiser** la machine a de nombreuses conséquences.
1. **Raisonnement sur les programmes** : on peut s'abstraire des détails de l'électronique et se donner un cadre formel et universel (indépendant du matériel) pour **prouver** des propriétés sur l'évaluation des programmes.
 2. **Contrôle de l'exécution de la machine** : le programme d'émulation de la machine peut **observer un programme avant** de l'évaluer pour vérifier qu'il se comporte correctement. Une machine virtuelle peut aussi rattrapper les erreurs, adapter un programme a son environnement, etc.
 3. **Manipulation de la machine elle-même comme une donnée** : on peut faire transiter une machine virtuelle à travers internet, la mettre à jour, etc.

Les machines virtuelles : le revers de la médaille

- | L'implémentation de machines virtuelles **efficaces** et **sûres** est difficile.
 - | Le problème d'**efficacité** provient de la **couche d'interprétation** introduite par la machine virtuelle.
 - | Le problème de **sûreté** est intrinsèque à la difficulté d'analyser des programmes avant de les évaluer (analyse dite **statique**).
- ⇒ L'objectif de ce cours est l'étude des solutions modernes à ces problèmes en termes de logiciel et de matériel.

Quelques rappels sur les machines de Turing

La première de toute : la machine de Turing



© Pour la Science

Ingrédients :

- | Une **bande** (infinie) de cases contenant des symboles.
- | Une **tête de lecture** en face d'une case
- | Un **état courant** (p.ex. un numéro d'état)
- | Un **programme** (fini), qui à tout état et symbole lu, associe un nouvel état, un symbole à écrire, et un mouvement (G/D) de la tête de lecture.

Machine de Turing

1. Écrire une addition de bâtonnets : la bande devra passer de « $XX+XXXX$ » à « $XXXXXX$ »
2. Écrire une soustraction de bâtonnets.
3. (dur) Écrire une multiplication de bâtonnets.
4. Donner un exemple de machine ne s'arrêtant pas. Comment savoir à l'avance si on est dans ce cas ?

Machine de Turing et problème de l'arrêt

- | Création en 1936 (1er ordinateur : 1945!).
- | Sa motivation, l'**Entscheidungsproblem** :
Existe-t-il une machine répondant à toutes les questions oui/non des maths ?
- | Réponse négative, via l'indécidabilité du **problème de l'arrêt** : pas de méthode générale pour savoir si une machine et une bande initiale vont mener à un arrêt.
- | Ici, "méthode générale" = algorithme = machine de Turing
- | Ces égalités forment la **Thèse de Church-Turing** : tout ce qui se calcule (même en Java 1.5 sur Amd 64 3500+) peut se faire *via* une machine de Turing.

Machine de Turing, Variantes et Ordinateurs

- | Il existe de nombreuses **variantes** de la machine de Turing : alphabet des symboles, bandes multiples, bande bornée d'un côté. La plupart sont **équivalentes**.
- | Remarque sur la bande **infinie** : c'est juste une convenance.
Un programme ne peut utiliser une infinité de bande en un temps fini ! Ici, infinie = aussi large que nécessaire.
- | Analogie avec un ordinateur :
 - ▶ bande = mémoire
 - ▶ état = pointeur de code (pc)
 - ▶ programme = programme (ou plutôt câblage du CPU)

Machine de Turing universelle

- | Une seule machine suffit pour tout calculer !
 - | Astuce : on cache un **codage** d'une M.d.T. dans les données de la machine universelle, qui ensuite simule cette M.d.T.
 - | Autrement dit : on peut écrire en M.d.T un **interprète** de M.d.T...
 - | Dix ans avant le premier ordinateur polyvalent (encore que, cf Jacquard, Babbage, Pascal...).
- ⇒ Plus de détails ? cours de M1 Calculabilité et Complexité ...

Implémentation des machines

Méthodes d'implémentation

Spécification S d'une machine abstraite

+
Programme P écrit pour S \Rightarrow Résultat R
+
Entrée I de P

Le problème de l'évaluation : Comment calculer R ?

- | L'interprétation : on réalise S par un programme *Interprete*, alors :

$$R = \text{Interprete}(P, I)$$

(Mais qui évalue le programme *Interprete* ?)

- | La compilation : on traduit le programme P en un programme P' **équivalent** que l'on sait déjà évaluer :

$$R = P'(I)$$

Comment interpréter efficacement ?

- | Le sujet de ce cours **l'implémentation efficace** des interprètes.
- | Trois grandes directions :
 1. Des techniques d'écriture d'interprète.
 2. Le choix de langages simples à interpréter.
 3. La virtualisation.

Comment écrire un interprète ?

- | Comment écrire un interprète de OCaml en OCaml ?
 - | Trois ingrédients :
 - ▶ un type **expression** mimant les constructions du langage ;
 - ▶ un type **value** des résultats possibles (int/bool/fun) ;
 - ▶ un **environnement** associant à chaque variable sa valeur.
- (cf. le code)
- | Bilan : **simple**, **portable** mais **très lent**.

Comparaison avec le code « natif »

- | Le code « natif » est le langage interprété par la machine hôte.
- | C'est le moyen le plus efficace de calculer sur cette machine.
- | Regardons de plus près le code binaire à l'aide d'objdump.
- | Bilan : **non portable** mais **très rapide**.

Un code binaire portable : le code-octet

- | Les machines virtuelles des langages OCaml, Java ou CLR sont des **approximations** raisonnables des machines physiques existantes.
- | Leur proximité avec les architectures matérielles sur lesquelles elles sont exécutées permet de **réduire au minimum la couche d'interprétation** (*i.e.* le travail de l'interprète).
- ⇒ Nous étudierons précisément l'implémentation de la VM de Java et d'OCaml.
 - | On peut même totalement **supprimer** la couche d'interprétation à l'aide de techniques de **compilation** appliquées juste avant l'évaluation.
- ⇒ Nous parlerons de ces techniques utilisées dans la JVM et dans des émulateurs.
 - | Bilan : **relative simplicité, portabilité et rapidité raisonnable.**

Comment interpréter en toute sécurité ?

Le code-octet, une donnée comme une autre ?

- | Le code-octet est une application directe de l'architecture de Von Neumann : le programme peut être vu comme une donnée quelconque.
- | Cependant, en pratique, si on télécharge une mise-à-jour sur sa carte bancaire, qui nous garantit qu'elle ne mettra pas à mal l'intégrité de notre compte en banque ?
- ⇒ En tant que **donnée exécutable**, le code-octet doit être manipulé avec précaution.
- | Les solutions connues :
 - ▶ l'authentification de l'origine du code-octet (certificats, etc.) ;
 - ▶ l'analyse statique : on rejette les programmes dont on n'a pas pu prouver automatiquement le bon comportement.
 - ▶ la *sandbox* (bac à sable) : on virtualise la machine hôte en contrôlant les accès aux informations et périphériques critiques.

Fonctionnement du cours

Un cours/TD

- | Horaire : mardi de 9h à 11h.
 - | Le cours sera assuré par Pierre Letouzey.
 - | Le TD sera assuré par Thibaut Balabonski.
 - | Évaluation :
 - ▶ 1ere session : 2/3 examen + 1/3 c.c.
 - ▶ 2eme session : max (examen, 2/3 examen + 1/3 c.c.)
- Le contrôle continu sera un TP noté.
- | Page du cours : <http://www.pps.jussieu.fr/~letouzey>.
- ⇒ Inscrivez-vous sur le Didel MV6 !

Bibliographie

- | Cours atypique, donc pas de livre “tout-en-un”.
- | Sur les **machines de Turing** :
Calculabilité et décidabilité
Jean-Michel Autebert. Masson, 1992.
- | Sur **CamL** :
Développement d'applications avec Objective Caml
E. Chailloux, P. Manoury, et B. Pagano, Éditions O'Reilly
www.pps.jussieu.fr/Livres/ora/DA-OCAML/index.html
- | Voir aussi : cristal.inria.fr/~lebotlan/docaml_html et
<http://cadmium.x9c.fr/distrib/caml-instructions.pdf>
- | Sur **Java** :
The Java Virtual Machine
J. Meyer et T. Downing
- | **Généralités** :
Virtual Machines
Ian D. Craig, Springer

Synthèse

QCM : Question 1

Parmi ces systèmes, lesquels utilisent des modèles de calcul **séquentiels** ?

- ☐ Les processus chimiques
- ☐ Les processus biologiques
- ☐ Du code binaire *i386*
- ☐ Une requête SELECT de SQL

QCM : Question 2

Une machine virtuelle :

- ☐ est l'émulation logiciel d'une machine abstraite
- ☐ exécute toujours plus rapidement les programmes que sa machine hôte
- ☐ est irréelle
- ☐ s'implémente toujours en assembleur

QCM : Question 3

À l'aide d'une machine virtuelle, on peut

- ☐ s'assurer que seuls des programmes inoffensifs sont exécutés.
- ☐ s'assurer que seuls les programmes dangereux ne seront pas autorisés à s'exécuter.
- ☐ faire fonctionner un programme écrit en code *i386* sur une machine *PPC*.
- ☐ évaluer un programme quelque soit le langage dans lequel il est écrit.

QCM : Question 4

Un interprète, c'est

- ☐ la même chose qu'un compilateur.
- ☐ un programme qui exécute un autre programme.
- ☐ par exemple, un micro-processeur.
- ☐ par exemple, la machine virtuelle JAVA.

QCM : Question 5

Java est un langage :

- ☐ interprété uniquement.
- ☐ compilé et interprété.
- ☐ de programmation qui a plusieurs implémentations.

QCM : Question 6

Une analyse statique

- ☐ exécute un programme une première fois avant l'exécuter pour de bon.
- ☐ sert à exhiber des propriétés de l'évaluation d'un programme.
- ☐ ne peut pas répondre à toutes les questions sur tous les programmes.