Structures de contrôle

Introduction et plan du cours

Xavier Leroy

2024-01-25

Collège de France, chaire de sciences du logiciel
xavier.leroy@college-de-france.fr

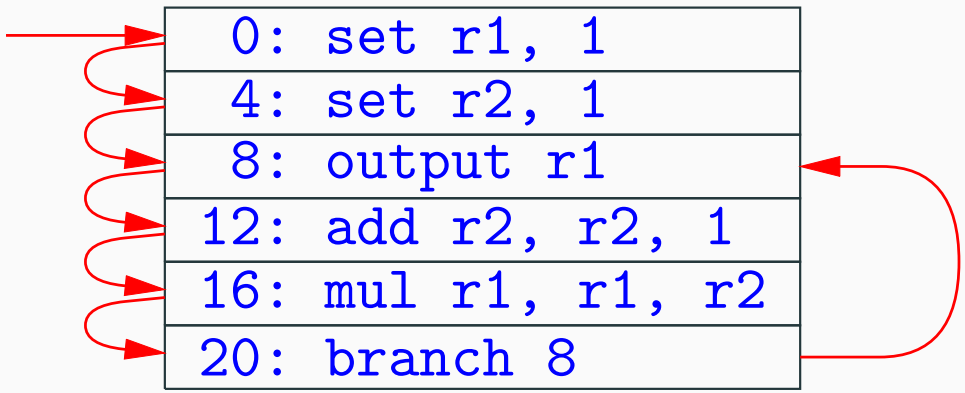
Comment décrire et contrôler l'enchaînement des opérations ?

Par exemple : avec des cartes perforées mises en boucle !

Le contrôle dans les architectures matérielles des ordinateurs

Un modèle prédominant depuis 1945 (von Neumann, Turing) :

- la plupart des instructions sont exécutées en séquence, dans l'ordre où elles sont placées en mémoire ;
- les instructions de branchement permettent de ≪sauter≫ à une adresse de code spécifique.



Le contrôle dans les langages de programmation

Les premiers langages suivent le modèle de la machine : contrôle non structuré avec branchements — le célèbre `goto`.

Les langages évoluent ensuite vers un contrôle structuré exprimé à l'aide de mécanismes linguistiques appelés structures de contrôle :

- conditionnelles, boucles comptées, boucles générales, …
- procédures, fonctions, méthodes, coroutines, …
- gestion d'exceptions, de continuations, d'effets.

D'autres langages cherchent à rendre le contrôle entièrement implicite : les langages déclaratifs
(*dataflow*, fonctionnel, logique, par contraintes, …)

Ce cours : les paradigmes du contrôle

Un voyage dans l'espace et le temps des langages de programmation, vus sous l'angle du contrôle :

- quelles constructions fournissent-ils aux programmeurs pour exprimer et contrôler l'enchaînement des calculs ?
- quelle structuration du programme (code source) est rendue possible et encouragée par ces constructions ?

Une approche descriptive, mais surtout comparative, et parfois formelle.

Plan du cours

25/01 Naissance des structures de contrôle : du ≪ goto ≫ à la programmation structurée.

- Assembleur, FORTRAN, Algol, et descendants.
- Le mouvement pour la programmation structurée.
- Expressivité du contrôle structuré.

01/02 Le contrôle non local : des subroutines aux fonctions et aux coroutines.

- Subroutines, procédures, fonctions : structuration et réutilisation.
- Le `goto` non local vs. les exceptions.
- Itérateurs, générateurs, coroutines, *threads*.

08/02 Chassez le contrôle… : la programmation déclarative

- Langages d'expressions, langages *dataflow*.
- Langages fonctionnels.
- Langages logiques et par contraintes.

Plan du cours

15/02 Programmer ses structures de contrôle : continuations et opérateurs de contrôle.

- Les continuations comme outil sémantique.
- Programmation en style à passage de continuation.
- Opérateurs de contrôle ; continuations en style direct.

22/02 Pratique des effets : des exceptions aux gestionnaires d'effets.

- Exceptions, exceptions redémarrables, capture de continuation.
- Les gestionnaires d'effets en OCaml 5.
- Exemples d'utilisation : coroutines, *threads*.

29/02 Théorie des effets : des monades aux effets algébriques.

- Les monades : une théorie de la propagation des effets.
- Les effets algébriques : une théorie de la génération et du traitement des effets.
- Gestionnaires d'effets et arbres d'interaction.

Plan du cours

07/03 Typage et analyse statique des effets

- Systèmes de types et d'effets.
- Analyses statiques : *k*-CFA, *set-based analysis*.

14/03 Logiques de programmes pour le contrôle et les effets

- Logiques de Hoare pour les structures de contrôle classiques.
- Logiques de séparation pour les coroutines et les générateurs.
- Logiques de séparation pour les gestionnaires d'effets.

Plan du séminaire

08/02 Caroline Collange (Inria Rennes)
Comment concilier parallélisme et contrôle ? Approches des architectures de processeurs généralistes et graphiques.

15/02 Delphine Demange (U. Rennes)
Représentations intermédiaires pour la compilation : s'affranchir du graphe de flot de contrôle.

22/02 Andrew Kennedy (Meta)
Compiling with Continuations.

29/02 Olivier Danvy (National U. of Singapore)
Les continuations : cinq minutes pour les apprendre, toute une vie pour les comprendre.

Plan du séminaire

07/03 Matija Pretnar (U. Ljubljana)
How mathematics guides effect handlers.

14/03 Daan Leijen (Microsoft Research)
Design and Compilation of Efficient Effect Handlers in the Koka Language.