École normale supérieure de Lyon

Internship Report

CFG Patterns: A new tool to formally verify optimisations in Vellvm

Leon Frenot

supervised by
Yannick Zakowski & Gabriel Radanne
at ENS Lyon

February 5th, 2024 - July 5th, 2024

Contents

Abstract

Abstract

1 Introduction

Debut intro: M2, 20 semaines, LIP, CASH. Yannick Zakowski & Gabriel Radanne. Goal.

Compilation certifiée AJD

Importance de la compilation certifiée, et surtout de certifier les optims.

The Contribution of This Work

- Design d'un langage de patterns + Implémentation naive d'un matcher
- Preuve d'un théorème central pour prouver des optims (sur un CFG)
- Utiliser ce langage pour deux optims + preuves de correction

Premier exemple: CCstP

2 Key concepts

2.1 LLVM and Vellvm

llvm (très rapide)

vellvm: but, niveaux d'interprétation (préciser celui auquel on se place)

- denotational proofs, programmes ouverts → utilisera OCFG pour open CFG
- structure en couche, optimisations qui conservent les traces d'interaction

pourquoi travailler sur vellvm

2.2 ITrees

utilité, coinduction, structure, mechanisme de preuve

3 Le langage de patterns

- goal of section: define modular set of simple patterns that can capture optimizable subgraphs of an OCFG.
- capture: have a function that, given a pattern and an ocfg, returns a/the subgraph(s) corresponding to the pattern's grammar.

3.1 présentation des patterns + utilité

schéma pour chaque pattern
dire que head et branch sont déja des patterns “optimizés” → explications dans 7

- Graph: Pattern ocfg
- When: ∀ S, Pattern S → (S → bool) → Pattern S
- Head: ∀ S, Pattern S → Pattern (bid * blk * S)
- Focus: ∀ S, Pattern S → Pattern (ocfg * S)
- Map: ∀ S T, Pattern S → (S → T) → Pattern T
- Block: ∀ S, Pattern S → Pattern (bid * blk * S)
- Branch: ∀ S, Pattern S → Pattern (bid * blk * S)

3.2 présentation de patterns/optims possibles (non implémentés)

trouver optim avec map?

4 Cas d’étude: Block Fusion

4.1 motivation for Block Fusion

- blocks leftover by other optims
- modifies CFG
- fairly simple optim

4.2 Block Fusion pattern

defis d’interprétation (φ & term), renomage

5 Théorème: `denote_ocfg_equiv`

exemple plus précis d’une preuve par coinduction

intro

5.1 defis

présentation des défis que pose le formalisme + le niveau d’interprétation
schéma idée de base → problèmes noms et phi → hypothèses etc.

5.2 hypothèses

- hyp principale: `denote_ocfg_equiv_cond`
- hyps sur nTO et nFROM ! Schémas
- `dom_renaming`

5.3 lemmes

- `bk_phi_rename_eutt`

6 implémentation (+ raison pour s'arrêter à naïve (rapide))

7 A voir: Approfondissements

7.1 Loop pattern

7.2 Optim efficace

Conclusion