Freie Universität Berlin

Softwareprojekt Übersetzerbau
Optimierungstechniken

David Knötel, Björn Karger, Daniel Marzin
Freie Universität Berlin, Institut für Informatik

Ziele

- Gesucht: Sinnvolles Maß für den Abstand zweier Kurven $f : [a_1, b_1] \to \mathbb{R}^n$ und $g : [a_2, b_2] \to \mathbb{R}^n$
- Lösung: Einführung des Fréchet-Abstandes

Freie Universität Berlin

UML

Implementierung

- LLVM ist SSA
- f ist Kurve mit p Punkten $p_1, \ldots, p_p$, so dass $f : [1, p] \rightarrow \mathbb{R}^n$ mit $f(1) = p_1, \ldots, f(p) = p_p$ gilt und dazwischen linear interpoliert wird
- g ist Kurve mit q Punkten $q_1, \ldots, q_q$, so dass $f : [1, q] \rightarrow \mathbb{R}^n$ mit $g(1) = q_1, \ldots, g(q) = q_q$ gilt und dazwischen linear interpoliert wird

Eingesetzte Optimierungsalgorithmen

- Entscheidungsproblem: Ist der Fréchet-Abstand zweier Kurven kleiner gleich einem gegebenen ϵ?
- Lösung mit Hilfe des Free Space Diagram
- Definition:

$$F_\epsilon(f,g) = \{(s,t) \in [1,p] \times [1,q] : \|f(s) - g(t)\| \leq \epsilon\}$$

Optimierungen & Beispiele

- Constant folding

Test1

unoptimierter Code

Test1

optimierter Code

Optimierungen & Beispiele

- Constant folding
- Constant propagation

Test2

unoptimierter Code

Test2

optimierter Code

Optimierungen & Beispiele

- Constant folding
- Constant propagation
- Store/Load folding

Test3

unoptimierter Code

Test3

optimierter Code

Optimierungen & Beispiele

- Constant folding
- Constant propagation
- Store/Load folding
- Remove common expressions

Test4

unoptimierter Code

Test4

optimierter Code

Optimierungen & Beispiele

- Constant folding
- Constant propagation
- Store/Load folding
- Remove common expressions
- Global life time analysis

Test5

unoptimierter Code

Test5

optimierter Code

Optimierungen & Beispiele

- Constant folding
- Constant propagation
- Store/Load folding
- Remove common expressions
- Global life time analysis
- Strength reduction

Test6

unoptimierter Code

Test6

optimierter Code

Optimierungen & Beispiele

- Constant folding
- Constant propagation
- Store/Load folding
- Remove common expressions
- Global life time analysis
- Strength reduction
- Eliminate dead registers/blocks

Test7

unoptimierter Code

Test7

optimierter Code

Fazit / Ausblick

- Es wurden mehrere Optimierungsalgorithmen erfolgreich auf einen gegebenen LLVM-Code angewendet. Auch bei natürlichem, aus C-Code über CLANG erzeugten LLVM-Code wurden erfolgreich bedeutende Mengen an Codezeilen entfernt.

- Eine Weiterentwicklung des Programms wäre durch die Vielzahl an potentiellen weiteren Optimierungstechniken problemlos möglich. Priorität hätte hierbei die Anwendung von Schleifenoptimierungen.

Danke

Vielen Dank für Ihre Aufmerksamkeit.

Quellen

- Aho, Alfred V. ; Lam, Monica S. ; Sethi, Ravi; Ullman, Jeffrey D.: Compilers: Principles, Techniques and Tools.
- LLVM Language Reference Manual. http://llvm.org/docs/LangRef.html (Abruf 17.07.2012).