LVA 185.A04 Optimierende Compiler (WS 20)

Leit- und Kontrollfragen XI

Di, 15.12.2020

Stoff: Vorlesungsteil IV – Kapitel 14 und 15

Erweiterungen: Alias-Analysen, Optimierungen für objektorientierte Sprachen

(Ohne Abgabe, ohne Beurteilung; zur Selbsteinschätzung)

Teil IV, Kapitel 14 'Alias-Analysen'

1. Was ist im Zusammenhang mit Programmanalyse mit Aliasen gemeint?
2. Wie können Aliase entstehen? Was sind wichtige Quellen dafür?
3. Wie unterscheiden sich (die Ziele einer)
   (a) Zeiger-
   (b) Alias-
   (c) Form- (engl. shape)

   Analyse voneinander?
4. Welche zwei wichtigen Spielarbeiten sind für Aliasanalysen zu unterscheiden?
5. Welche Konstrukte in Programmiersprachen machen die Formulierung und Implementierung von Alias-Analysen besonders anspruchsvoll?
6. Geben Sie Beispiele an, wie Alias-Information für Programmoptimierungen ausgenutzt werden kann.
7. Wie unterscheiden sich
   (a) (kontroll-) flusssensitive
   (b) (kontroll-) flussinsensitive

   Alias-Informationen voneinander?
8. Warum sind (kontroll-) flusssensitive und -insensitive Alias-Informationen beide nützlich?
9. Alias-Informationen werden meist in Form von
   (a) Graphen
   (b) regulären Ausdrücken
   (c) Ausdrücken einer dreiwertigen Logik

   repäsentiert. Erläutern Sie diese Darstellungsformen anhand von Beispielen.
10. Warum sind zur Darstellung von Alias-Informationen Ausdrücke einer dreiwertigen Logik besser geeignet als die einer zweiwertigen? Sind Ausdrücke einer zweiwertigen Logik überhaupt dafür geeignet?
11. Was versteht man unter starken, was unter schwachen Aktualisierungen (engl. strong/weak updates)?
12. Was sind typische Fragen, die mittels geeigneter Programmanalysen über die von einem Programm auf dem Haufenspeicher verwalteten Daten beantwortet werden sollen?
13. Was ist das Ziel einer Form-Analyse (engl. shape analysis)?
14. Was sind typische Formeigenschaften, die man herausfinden möchte über
    (a) Datenwertstrukturen
    (b) datenwertstrukturmanipulierenden Prozeduren, Funktionen?
15. Auf wen gehen die ersten Verfahren zu Formanalyse zurück?

Teil IV, Kapitel 15 'Optimierungen für objektorientierte Sprachen'

1. Welche Optimierungen liegen im Zusammenhang mit dem Aufruf von Methoden nahe?
2. Was sind hinreichende Bedingungen ($\widehat{=}$ Informationen) dafür, diese Optimierungen in einem Programm anzuwenden?]
3. Welche anderen Optimierungen sind damit verbunden?
4. Bei welchen dieser Optimierungen handelt es sich im engeren Sinn um Optimierungen, bei welchen um eher um 'nicht naive' Übersetzung? Begründen Sie Ihre Antwort.
5. Was sind Beispiele typischer Analysen zur Auflösung (engl. devirtualization) virtueller Methodenaufrufe?
6. Wie spielen Objektspeicherorganisation und Methodenaufrufe zusammen?
7. Welche Speicherorganisation für Objekte wird im Fall
   (a) einfacher
   (b) mehrfacher

   Vererbung oft gewählt? Illustrieren Sie die Speicherorganisation anhand geeigneter Beispiele.
8. Skizzieren Sie die Verfahrensideen von:
   (a) Klassenhierarchieanalyse
   (b) Schneller Typanalyse (engl. rapid type analysis)
   (c) Einkopieren (engl. inlining).
9. Welche Rolle spielt der sog. Aufrufgraph (engl. call graph) für die Klassenhierarchieanalyse?
10. Welche wichtigen Spielarten Schneller Typanalyse lassen sich unterscheiden?
11. Skizzieren Sie die Unterschiede dieser beiden Spielarten und ihre relativen Vor- und Nachteile.
12. Skizzieren Sie das Zusammenspiel der Auflösung virtueller Methodenaufrufe und dem EInkopieren von Methoden. Von welchen Optimierungseffekten wird profitiert?
13. Skizzieren Sie die Idee von Fluchtanalyse (engl. escape analysis).
14. Wer oder was flüchtet? Was sind Fluchtquellen?
15. Wie kann Fluchtinformation zur Programmoptinierung ausgenutzt werden?
16. Wozu dienen Konnektionsgraphen? Was modellieren sie? Was beschreiben die Knoten von Konnektionsgraphen? Was die Kanten?