LVA 185.276 Analyse und Verifikation (SS 2020)

Selbsteinschätzungstest 9

Mo, 11.05.2020

Stoff: Vorlesungsteil IV, Kapitel 13 und 14

Transformationskombinationen, nichtklassische Programm- und Datenstrukturen für Analyse und Optimierung

(Ohne Abgabe, ohne Beurteilung)

Teil IV, Kapitel 13 'Transformationskombinationen'

- Die Optimalitätsaussagen in Kapitel 13.1 garantieren Eindeutigkeit der Transformationsergebnisse "bis auf irrelevante Umsortierungen in Basisblöcken".

  1. Was ist damit gemeint? Geben Sie ein Beispiel zur Veranschaulichung an.
  2. Welche andere Darstellung von Basisblöcken würde die Einschränkung "eindeutig bis auf..." obsolet machen? Denken Sie an Datenabhängigkeitsgraphen.

- Im Zusammenhang mit Programmtransformationen zur Beseitigung unnötiger Anweisungen in Programmen werden manche Kanten eines Flussgraphen als "kritisch" bezeichnet.

  3. Welche Kanten sind das? Welche Eigenschaft erfüllen sie?
  4. Warum ist die Bezeichnung 'kritisch' für diese Kanten gerechtfertigt? Was verhindern sie möglicherweise?
  5. Wie können die Probleme, die kritische Kanten verursachen können, verhindert werden?

- Die Drei-Adress-Form ist eine bestimmte Form von Programmdarstellungen.

  6. Wann liegt ein Programm in Drei-Adress-Form vor?
  7. Kann jedes Programm in Drei-Adress-Form überführt werden, dargestellt werden?

- Oft finden sich Aussagen wie "ohne Beschränkung der Allgemeinheit betrachten wir Programme in Drei-Adress-Form".

  8. Was ist damit gemeint?
  9. Ist diesen Aussagen zuzustimmen? Ist ihnen uneingeschränkt zuzustimmen?
  10. Gibt es Fälle, in denen genauer hingeschaut werden muss? Veranschaulichen Sie Ihre Antwort ggf. anhand von Beispielen.

- Ob Programme als Basisblock- oder EInzelanweisungsgraphen dargestellt werden, wird im Sinn eines "ohne Beschränkung der Allgemeinheit" ebenfalls oft so oder so entschieden.

  11. In welchem Sinn ist das gerechtfertigt?
  12. Gibt es Gründe oder Fälle, an einigen Stellen genauer hinzuschauen? eranschaulichen Sie Ihre Antwort ggf. anhand von Beispielen.

- Datenflussanalyseprobleme können in vielerlei Hinsicht geordnet und klassifiziert werden. U.a. wird so zwischen separablen und nichtseparaben Problemen unterschieden.

  13. Illustrieren Sie den Unterschied zwischen separablen und nichtseparablen Datenflussanalyseproblemen anhand der Analysen für tote und Geistervariablen.
  14. Was sind andere Beispiele für separable und nichtseparable Datenflussanalyseprobleme?

– Die EPTRA-Programmtransformation ist Verallgemeinerung zweier anderer Grundtransformationen.

15. Welche sind das?
16. Wie lassen sich diese Transformationen durch regulär-ähnliche Ausdrücke beschreiben?
17. Welche Rolle spielen Konfluenz und Terminierung für diese Transformationen?
18. Was ist intuitiv von der Verallgemeinerung gegenüber den Grundtransformationen zu erwarten?
18. Bestätigen sich diese Erwartungen in der Praxis?
20. Bestätigen sich diese Erwartungen formal, in Form von Optimalitätsaussagen?

Teil IV, Kapitel 14 'Nichtklassische Programm- und Datenstrukturen für Analyse und Optimierung'

1. Was versteht man unter SSA-Form eines Programms? Wofür steht das Akronym SSA?
2. Was ist der Wertegraph eines Programms? Auf wen geht diese Datenstruktur zurück?
3. Auf welche Weise sind SSA- und Wertegraph eines Programms miteinander verbunden?
4. Welche Konstantenklasse erkennt das Grundverfahren zur Konstantenanalyse auf dem Wertegraphen?
5. Woran wird mit dem "Triple E – Rating" begriffstechnisch eine Anleihe gemacht? Was ist damit gemeint?
6. Auf welche Weise erlauben ϕ-Konstanten, dem vollen Algorithmus auf dem Wertegraphen eine umfassendere Klasse von Konstanten als einfache Konstanten zu erkennen?
7. Was ist prädikatierter Code?
8. Was ist der ursprüngliche Grund gewesen, prädikatierten Code von Prozessoren unterstützen zu lassen?
9. Was sind neue Probleme, vor die prädikatierter Code Programmanalyse im Vergleich zu unprädikatiertem Code stellt?
10. Wie lassen sich diese Probleme anhand von Konstantenanalyse anhand von Beispielen aufzeigen?

Teil I – IV, Verschiedene Kapitel

1. Erweitere den Hoare-Kalkül für partielle Korrektheit um eine Regel zur Behandlung der repeat-Schleife:

   `repeat` π `until b end`

   ohne sich dabei auf die Existenz der while-Schleife und deren entsprechende Hoare-Kalkülregel abzustützen.

2. Beweise mit der Regel aus der vorigen Aufgabe die Korrektheit des Hoare-Tripels:

   $$\{x > 0\}\ \ z := 0;\ u := x;\ \textbf{repeat}\ z := z + y;\ u := u - 1\ \textbf{until}\ u = 0\ \textbf{end}\ \ \{z = x * y\}$$

3. Wie lässt sich die Richtigkeit der Regel aus Aufgabenteil 1 zeigen? Was ist zu beweisen?
4. Welche Bedeutung hat das beobachtbare Verhalten von Programmen auf den Entwurf von Programmtransformationen?
5. Verbessernde Programmtransformationen, sog. (Programm-) Optimierungen, sollen korrekt und vollständig sein. Was ist damit gemeint?

6. Welche weiteren Eigenschaften von Programmtransformationen sind darüberhinaus wünschenswert?

7. Einige Programmtransformationen sehen vor, dass kritische Kanten aufgespalten sind. Was ist damit gemeint? Warum ist es für einige Transformationen sinnvoll oder nötig?

8. Illustrieren Sie die Grundmuster der Beseitigung toter und geisterhafter Anweisungen anhand geeigneter Beispiele.

9. Welche Optimalitätsergebnisse gelten für die Beseitigung partiell toter und geisterhafter Anweisungen?

10. Erklären Sie die Intuition, die dem Gleichungssystem zur Berechnung toter Variablen zugrundeliegt:

$$\mathsf{N\text{-}DEAD}_n^v = \overline{\mathsf{Use}_n^v} * (\mathsf{X\text{-}DEAD}_n^v + \mathsf{Mod}_n^v)$$

$$\mathsf{X\text{-}DEAD}_n^v = \prod_{m \in succ(n)} \mathsf{N\text{-}DEAD}_m^v$$

11. Was besagt das Fixpunkttheorem von Knaster, Tarski und Kleene? Können Sie es aus dem Stand beweisen? In welche Teilschritte könnten Sie den Beweis zerlegen?

12. Geben Sie die Komponenten des reversen Problems zur Erkennung toter Anweisungen im Detail an.

13. Die Bedeutung einer Programmiersprache ist in Form einer natürlichen Semantik definiert. Gültigkeitsbeweise von Programmeigenschaften werden in der Regel dann wie geführt?

14. Die Unterscheidung zwischen Verbänden und vollständigen Verbänden ist nur über unendlichen Grundmengen sinnvoll. Richtig oder falsch? Begründen Sie Ihre Antwort.

15. Welcher Semantikdefinitionsstil spricht besonders den Implementierer einer Programmiersprache an. Warum?

16. Was unterscheidet eine kettenvollständige partielle Ordnung von einem Verband, was von einem vollständigen Verband?

17. Was unterscheidet eine Quasiordnung, eine partielle Ordnung, eine Äquivalenzrelation voneinander?

18. Was ist ein Hasse-Diagramm? Wozu dienen Hasse-Diagramme? Geben Sie auch ein Beispiel eines Hasse-Diagramms an, um Ihre Antwort zu illustrieren.

19. Wie lässt sich Monotonie einer Funktion auf vollständigen Verbänden äquivalent ausdrücken, um den Unterschied zu distributiven und additiven Funktionen besonders augenfällig zu machen?

20. Warum ist die absteigende Kettenbedingung in Verbänden für Datenflussanalyse wichtig? Warum die aufsteigende?