LVA 185.276 Analyse und Verifikation (SS 2020)

Selbsteinschätzungstest 7

Mi, 22.04.2020

Stoff: Vorlesungsteil III, Kapitel 9 und 10

Parallele Datenflussanalyse; Programmanalyse, Programmverifkation: Datenflussanalyse, axiomatische Verifikation gegenübergestellt

(Ohne Abgabe, ohne Beurteilung)

Teil III, Kapitel 9 'Parallele Datenflussanalyse'

1. Was bezeichnet das *Pfad-* bzw. *Zustandsexplosionsproblem*?
2. In welcher Weise treten das *Pfad-* bzw. *Zustandsexplosionsproblem* bei parallelen Programmen auf?
3. Welches sind die zentralen Probleme, die beim Übergang von der Analyse sequentieller zur Analyse paralleler Programme gelöst werden müssen?
4. Warum können das *Pfad-* und *Zustandsexplosionsproblem* für Bitvektoranalysen paralleler Programme vermieden werden?
5. Was sind sog. *kontrollflussfreie* Datenflussanalysen?
6. Welche Bedeutung hat kontrollflussfreie Analyse für Bitvektoranalysen paralleler Programme?
7. Worin liegt der Schlüssel für die Effizienz von Bitvektoranalysen für parallele Programme? Funktioniert der Schlüssel auch für andere Problemklassen?
8. Bitvektoranalysen sind einfach vom sequentiellen auf parallele Programme zu übertragen. Für auf Bitvektoreigenschaften aufbauende Programmtransformationen und -optimierungen gilt das auch. Richtig oder falsch? Begründe die Antwort.
9. Was ist der funktionale denotationelle Semantikansatz? Welche Rolle spielt er für die Analyse parallleler Programme?
10. Im welchem Zusammenhang steht die funktionale denotationelle Semantik zur denotionellen Semantik aus Kapitel 7? Welche Resultate gelten und verbinden die beiden?
11. Was ist der *Rang* eines parallelen Flussgraphen? Wofür dient er im Rahmen paralleler Datenflussanalyse?
12. Was sind *verschränkte* Vorgänger?
13. Was sind *trivial sequentialisierte* Flussgraphen?
14. Welche Rolle spielen sie im Rahmen paralleler Bitvektoranalysen?
15. Warum reicht es, sich für die hierarchische Analyse paralleler Flussgraphen für Bitvektorprobleme auf 'trivial sequentialisierte' Graphen abzust—ützen?

Teil III, Kapitel 10 'Programmanalyse, Programmverifkation: Datenflussanalyse, axiomatische Verifikation gegenübergestellt'

1. Skizziere den Zusammenhang und Analogien zwischen axiomatischer Programmverifikation und reverser Datenflussanalyse.
2. Wie lässt sich der Zusammenhang von Datenfluss- und reverser Datenflussanalyse graphisch veranschaulichen?
3. Illustriere Idee und Zusammenhang axiomatischer Programmverifikation aus Sicht stärkster Nachbedingungen und schwächster Vorbedingungen.

Teil I – III, Verschiedene Kapitel

1. Erweitere die Programmiersprache WHILE um das Schleifenkonstrukt

   `repeat` π `until b end`

   Definiere die Semantik der repeat-Schleife im Stil der natürlichen Semantik durch Angabe entsprechender NS-Regeln, ohne sich dabei auf die Existenz der while-Schleife und deren natürliche Semantik abzustützen; die repeat-Schleife soll dabei ihre "gewohnte" Bedeutung erhalten.

2. Zeige mithilfe der natürlichen Semantik von WHILE, dass die Programme π und π' von SET 6 angesetzt auf den Zustand $\sigma \in \Sigma$ mit $\sigma(x) = \mathbf{13}$ und $\sigma(y) = \mathbf{5}$ regulär terminieren und die Werte der Variablen x, y und z in den jeweiligen Endzuständen übereinstimmen.

3. In welcher konzeptuellen Weise unterscheiden sich denotationelle und axiomatische Semantik einer voneinander?

4. Was besagt das Prinzip der strukturellen Induktion?

5. Was wäre ein gutes Beispiel für eine Aussage, die mithilfe struktureller Induktion bewiesen würde?

6. Führe diesen Beweis.

7. Erläutere die denotationelle Semantik der while-Schleife.

8. Wie lässt sich das Sicherheitstheorem aus Kapitel 7 beweisen?

9. Spezifiziere als reverse Datenflussanalyse eine anforderungsgetriebene Datenflussanalyse, mit deren Hilfe bestimmt werden kann, ob eine bestimmte Variable an einem bestimmten Programmpunkt möglicherweise nicht initialisiert ist.

10. Was ist die Aufsammelsemantik einer Datenflussanalyse? Wie ist sie definiert?