LVA 185.A04 Optimierende Compiler (WS 20)

Leit- und Kontrollfragen VI

Di, 10.11.2020

Stoff: Vorlesungsteil II – Kapitel 6 bis Kapitel 8

Intraprozedurale DFA: Partielle Redundanzelimination, fleißiges, faules Codeverschieben

(Ohne Abgabe, ohne Beurteilung; zur Selbsteinschätzung)

Teil II, Kapitel 6 'Partielle Redundanzelimination'

1. Wie sieht das 'kanonische' Beispiel zur Illustration der Optimierungsidee von partiell redundanten Berechnungen aus?
2. Welche Optimierungsziele werden mit der Elimination redundanter Berechnungen verfolgt?
3. Welche Platzierungsstrategien hängen mit diesen Optimierungszielen zusammen?
4. Illustrieren Sie die Optimierungsziele und zugehörigen Platzierungsstrategien anhand geeigneter Beispiele.
5. Auf wen geht die Elimination partiell redundanter Berechnungen als Optimierungstechnik zurück?
6. Welche Optimierungstechniken wurden dadurch verallgemeinert (und weitgehend abgelöst)?
7. Wie alt (ungefähr) ist die Optimierungstechnik partiell redundanter Berechnungen?
8. Was hat diese Technik mit Gen/Kill-Problemen zu tun?
9. Welche Informationen werden für das Ursprungsverfahren zur Beseitigung partiell redundanter Berechnungen berechnet?
10. Gibt es eine Analyse, die gegenüber den anderen technisch eine Besonderheit darstellt? Welche und in welchem Sinn?
11. Das Ursprungsverfahren berechnet speziell die partielle Verfügbarkeit von Ausdrücken und darauf aufbauend eine als 'konstant' bezeichnete Eigenschaft für Knoten? Warum? Wozu dienen diese beiden Eigenschaften im Kontext des Gesamtverfahrens? Was sollen sie möglichst sicherstellen?
12. Gelingt dieses 'Sicherstellen'? Begründen Sie Ihre Antwort.
13. Welche Optimierungsziele erfüllt das Ursprungsverfahren für die Beseitigung partiell redundanter Berechnungen?
14. Welche Schwächen bzw. Beschränkungen hat dieses Verfahren aus
    (a) konzeptueller
    (b) (implementierungs-) technsicher

    Sicht?
15. Warum wohl wird die Beseitigung partiell redundanter Berechnungen auch als Codeverschiebung (code motion) bezeichnet?
16. Was versteht man im Zusammenhang mit der Beseitigung partiell redundanter Berechnungen als Codereplikation?
17. Was sind sog. kritische Kanten in einem Flussgraphen?
18. Welche Rolle spielen sie für die Erreichung berechnungsbester/-optimaler Ergebnisse?
19. Wie geht man im Kontext der Beseitigung partiell redundanter Berechnungen mit kritischen Kanten um?
20. Welche Art von Redundanzelimination hat Andrei P. Ershov in seinem CACM-Artikel von 1958 beschrieben?

Teil II, Kapitel 7 'Fleißiges Codeverschieben'– Busy Code Motion

1. Spalten kritischer Kanten oder spalten aller Kanten, die zu Knoten mit mehr als einem Vorgänger führen. Was spricht für das eine, was für das andere?
2. Auf welche beiden (Transformations-) Schritte lässt sich eine Codeverschiebungstransformation (CV-Transformation) konzeptuelll reduzieren?
3. Zulässige CV-Transformationen müssen
   (a) sicher
   (b) korrekt

   sein. Was bedeuten Sicherheit und Korrektheit hier?
4. Welche (übergeordnete) Eigenschaft ist für zulässige CV-Transformationen garantiert?
5. Welche drei lokalen Eigenschaften sind nützlich bzw. ausreichend, um die Eigenschaften Sicherheit und Korrektheit zu definieren?
6. Wofür sind die (generischen) Prädikatschreibweisen
   (a) $\mathit{Predicate}^{\forall}(p)$
   (b) $\mathit{Predicate}^{\exists}(p)$

   nützlich?
7. In welche Teileigschaften lässt sich die Sicherheitseigenschaft von CV-Transformationen aufspalten?
8. Zu welchen Gen/Kill-Eigenschaften sind diese Teileigenschaften äquivalent?
9. Welcher Zusammenhang besteht zwischen den Prädikaten Insert_{CM}, $\mathit{Replace}_{CM}$, *Safe* und $\mathit{Correct}_{CM}$ für zulässige Codeverschiebetransforamationen *CM*?
10. Was besagt das Sicherheitslemma? (Lemma 7.1.2.4)
11. Wie ist die Relation 'berechnungsbesser' auf der Menge zuässiger Codeverschiebetransformationen definiert?
12. Die Relation 'berechnungsbesser' ist eine partielle Ordnung auf der Menge zuässiger Codeverschiebetransformationen. Richtig oder falsch? Begründen Sie Ihre Antwort.
13. Wann heißt eine Codeverschiebetransformation berechnungsoptimal?
14. Nach Anwendung einer berechnungsoptimalen Codeverschiebetransformation sind Programme frei von partiell redundanten Berechnungen. Richtig oder falsch? Begründen Sie Ihre Antwort.
15. Wie kann die Platzierungsidee der fleißigen Codeverschiebetransformation informell beschrieben werden?
16. Beschreiben Sie informell die Aussagen des Frühestheitslemmas 7.2.3:
   (a) $\mathit{Safe}(n) \Rightarrow \forall\, p \in \mathbf{P}[s,n]\ \exists\, i \leq \lambda_p.\ \mathit{Earliest}(p_i) \wedge \mathit{Transp}^{\forall}(\,p[i,\lambda_p[\,)$
   (b) $\mathit{Earliest}(n) \iff \mathit{D\text{-}Safe}(n) \wedge \bigwedge_{m \in pred(n)} (\neg \mathit{Transp}(m) \vee \neg \mathit{Safe}(m))$
   (c) $\mathit{Earliest}(n) \iff \mathit{Safe}(n) \wedge \forall\, \mathcal{CM} \in \mathcal{CM}_{Adm}.\ \mathit{Correct}_{\mathcal{CM}}(n) \Rightarrow \mathit{Insert}_{\mathcal{CM}}(n)$
17. Beschreiben Sie informell die Aussagen des BCM-Lemmas 7.2.4:
   (a) $\forall\, i \leq \lambda_p.\ \mathit{Insert}_{BCM}(p_i) \iff \exists\, j \geq i.\ p[i,j] \in \mathit{FU\text{-}LtRg}(BCM)$
   (b) $\forall\, \mathcal{CM} \in \mathcal{CM}_{Adm}.\ \forall\, i,\, j \leq \lambda_p.\ p[i,j] \in \mathit{LtRg}(BCM) \Rightarrow \mathit{Comp}^{\exists}_{\mathcal{CM}}(\,p[i,j]\,)$
   (c) $\forall\, \mathcal{CM} \in \mathcal{CM}_{CmpOpt}.\ \forall\, i \leq \lambda_p.\ \mathit{Comp}_{\mathcal{CM}}(p_i) \Rightarrow \exists\, j \leq i \leq l.\ p[j,l] \in \mathit{FU\text{-}LtRg}(BCM)$
18. Welche Optimierungsziele partieller Redundanzelimination garantiert die fleißige Codeverschiebungstransformation? Welche nicht?
19. Illustrieren Sie Ihre Antwort zur vorigen Frage anhand geeigneter Beispiele.
20. Was gilt für den durch fleißige Codeverschiebung erzeugten Registerdruck?

Teil II, Kapitel 8 'Faules Codeverschieben – Lazy Code Motion'

1. Wie lässt sich informell die Verschiebestrategie fauler Codeverschiebung beschreiben?
2. Welche Optimierungsziele werden damit verfolgt?
3. Werden diese Ziele erreicht? Begründen Sie Ihre Antwort.
4. Welche Rolle spielen
   (a) Lebenszeitbereiche
   (b) Erste-Benutzung-Lebenszeitbereiche

   theoretisch und praktisch für faule Codeverschiebung?
5. Was sind
   (a) Lebenszeitbereiche
   (b) Erste-Benutzung-Lebenszeitbereiche?
6. Wann sprechen wir von trivialen Lebenszeitbereichen?
7. Lebenszeitbereiche sind entweder geschachtel oder disjunkt, aber nicht überlappend. Richtig oder falsch? Begründen Sie Ihre Antwort.
8. Die Relation 'lebenszeitbesser' ist eine partielle Ordnung auf der Menge der Codeverschiebetransformationen. Richtig oder falsch? Begründen Sie Ihre Antwort.
9. Was unterscheidet die faule und fast faule Codeverschiebetransformation?
10. Warum ist die Eigenschaft lebenszeitoptimal auf der Menge berechnungsoptimaler Codeverschiebetransformationen definiert, nicht auf der Menge aller Codeverschiebetransformationen oder der Menge zulässiger Codeverschiebetransformationen?
11. Warum ist die Einschränkung auf berechnungsoptimale Codeverschiebetransformationen für die Definition der Relation berechnungsbesser nicht nötig?
12. Was besagt das BCM-Lebenszeitbereichslemma informell?

$$\forall\, \mathcal{CM} \in \mathcal{CM}_{CmpOpt}.\ \forall\, p \in LtRg(\mathcal{CM}).\ \exists\, q \in LtRg(BCM).\ p \sqsubseteq q$$

13. Was besagt das Spätestheitlemma informell?
   (a) $\forall\, p \in LtRg(BCM)\ \exists\, i \leq \lambda_p.\ Latest(p_i)$
   (b) $\forall\, p \in LtRg(BCM)\ \forall\, i \leq \lambda_p.\ Latest(p_i) \Rightarrow \neg Delayed^{\exists}(\,p]i, \lambda_p]\,)$
14. Was verstehen wir im Zusammenhang mit fauler Codeverschiebung unter einer isolierten Berechnung?
15. Warum ist folgende Aussage des Isolationslemmas für die praktische Realisierung fauler Codeverschiebung so wichtig?

$$\forall\, \mathcal{CM} \in \mathcal{CM}_{CmpOpt}\ \forall\, n \in N.\ Latest(n) \Rightarrow (Isolated_{\mathcal{CM}}(n) \iff Isolated_{BCM}(n))$$

16. Das Ergebnis fauler Codeverschiebung ist für jedes Programm eindeutig bestimmt. Richtig oder falsch? Begründen Sie Ihre Antwort.
17. Welche Optimierungsziele partieller Redundanzelimination garantiert die faule Codeverschiebungstransformation? Welche nicht?
18. Illustrieren Sie Ihre Antwort zur vorigen Frage anhand geeigneter Beispiele.
19. Faule Codeverschiebung ist eine modulare Erweiterung fleißiger Codeverschiebung. Richtig oder falsch? Begründen Sie Ihre Antwort.
20. Was gilt für den durch faule Codeverschiebung erzeugten Registerdruck?