LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Prinzipien, Modelle und Algorithmen der Nebenläufigen Programmierung
Wintersemester 2019/20

Organisatorisches und Überblick

Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 27. November 2019

Beteiligte

Prof. Dr. David Sabel

- Raum L107, Oettingenstr. 67

David Tellenbach

- Aufgabenblätter
- Korrektur

Email-Verteiler an alle Organisatoren:

`concurrent-ws1920@tcs.ifi.lmu.de`

Termine

Vorlesung

- Donnerstag, 12-14, A 125 (Hauptgebäude)
- Freitag, 10-12 Raum 3232, (Leopoldstr. 13) ca. alle 2 Wochen

  Vorläufiger Plan für die Freitage:

18.10.2019	06.12.2019
25.10.2019	10.01.2020
08.11.2019	24.01.2020
22.11.2019	07.02.2020

Übung

- Dienstags 16-18, Lehrturm V005 (Prof. Huber-Pl. 2)
- Besprechung von Aufgaben

Webseiten / Material

- Durchklicken auf **www.tcs.ifi.lmu.de**
- Anmelden im uni2work!
- Aktuelle und organisatorische Informationen
- Unterlagen zur Veranstaltung: Skript, Folien, Aufgaben
- Referenzen auf Bücher, Webseiten, Programmiersprachen usw.

Aufgabenblätter

- Ungefähr: Jede Woche ein Aufgabenblatt
- Abgabe Dienstags bis 16 Uhr
- 1. Aufgabenblatt wird nicht abgegeben / korrigiert
- Abgabe und Korrektur über uni2work
- pro Blatt ca. jeweils 2 bepunktete Aufgaben mit 2 Punkten pro Aufgabe
- Bonus für die Prüfung

Hörerkreis und Prüfung

- Modul Algorithmik und Komplexität im Masterstudiengang Informatik
- 6 ECTS, 3V+2Ü

Modulprüfung

- Mündliche Prüfung oder Klausur
- „Bonus“ bei erfolgreicher Bearbeitung der Übungsaufgaben:
  - 100% erfolgreiche Bearbeitung aller Aufgaben
    $=$ 10% aller Klausurpunkte als Bonus
  - Anrechnung ansonsten linear
  - Bonuspunkte helfen nicht beim Bestehen
- bei mündlicher Prüfung entsprechende Würdigung (ab 50% erfolgreiche Bearbeitung eine Teilnote besser, bei ca 90% erfolgreiche Bearbeitung zwei Teilnoten besser)

Inhalt (Planung) (1)

Synchronisation von Prozessen und **Mutual-Exclusion**

- Mutual-Exclusion $=$ wechselseitiger Ausschluss
- Unter verschiedenen Annahmen der Speicheroperationen:
- Zunächst: Nur atomares read und write
- Mutual-Exclusion bei 2 Prozessen:
  Algorithmen von Dekker, Peterson, Kessels
- Mutual-Exclusion bei n Prozessen:
  Lamports Algorithmus, Bakery-Algorithmus

Inhalt (Planung) (2)

Stärkere Speicherobjekte

- Stärke Speicheroperationen und Mutual-Exclusion:
  Test-and-set Bits, RMW-Objekte, Compare-and-Swap Objekte
- Qualitative Klassifzierung der Speicheroperationen
- Consensus und die Consensuszahl

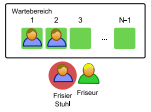
Inhalt (Planung) (3)

Gebräuchliche Konstrukte der Nebenläufigen Programmierung:

- Semaphoren
- Monitore
- Kanäle
- Tuple Spaces

Dabei: Lösung klassischer Probleme mit den Primitiven:

- Das Problem der Speisenden Philosophen
- Erzeuger / Verbraucher Probleme
- Sleeping Barbers Problem
- Cigarette Smoker's Problem
- Programmierung von Barrieren
- Das Readers & Writers Problem



Inhalt (Planung) (4)

Nebenläufiger Zugriff auf mehrere Ressourcen

- Deadlocks
- Deadlock-Vermeidung (Zwei-Phasen-Sperrprotokoll)
- Deadlock-Verhinderung (Bankiers Algorithmus)
- Transactional Memory (allgemein)

Inhalt (Planung) (5)

Nebenläufige und parallele Programmierung in Haskell
(evtl. auch weitere Programmiersprachen)

- Parallel Haskell
- Concurrent Haskell
- Software Transactional Memory in Haskell

Inhalt (Planung) (6)

Semantische Modelle nebenläufiger Programmiersprachen:

- Kurze Einführung in die Semantik von Programmiersprachen
- der π-Kalkül als Message-Passing-Modell
- der CHF-Kalkül als Shared-Memory-Modell
- Beziehungen zwischen den Modellen
- evtl. weitere Modelle

Literatur (1)

Zu Mutual-Exclusion-Algorithmen und Konstrukte der nebenläufigen Programmierung

Ben-Ari, M. (**2006**). *Principles of concurrent and distributed programming*. Addison-Wesley.

Taubenfeld, G. (**2006**). *Synchronization Algorithms and Concurrent Programming*. Prentice-Hall.

Reppy, J. H. (**2007**). *Concurrent Programming in ML*. Cambridge University Press.

Raynal, M. (**2013**). *Concurrent Programming: Algorithms, Principles, and Foundations*, Springer.

Literatur (2)

Zu Concurrent Haskell:

- **Peyton Jones, S. & Singh, S.** (**2009**). A tutorial on parallel and concurrent programming in haskell. In P. Koopman, R. Plasmeijer, & D. Swierstra, editors, *Advanced Functional Programming, 6th International School, AFP 2008, Revised Lectures*, volume 5832 of *Lecture Notes in Computer Science*, pages 267–305. Springer. ISBN-13 978-3-642-04651-3.
- **Marlow, S. (2013)**. *Parallel and Concurrent Programming in Haskell*, O'Reilly.

. . .

Literatur (3)

Zum π-Kalkül:

Milner, R. (**1999**). *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press.

Sangiorgi, D. & Walker, D. (**2001**). *The π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, New York, NY, USA.

Weitere Literatur

- Verweise auf speziellere Forschungsartikel etc. werden im Skript zu den einzelnen Themen angegeben.