

Entwurf und Implementierung paralleler Programme

Prof. Dr. Rolf Hennicker

16.10.2006

Kapitel 1

Einführung

1.1 Begriffsbildung

Sequentielles Programm:

Anweisungen werden Schritt für Schritt hintereinander ausgeführt (“single thread of control”).

Paralleles Programm (nebenläufig, concurrent):

Anweisungen (von Teilen des Programms) werden nebeneinander ausgeführt (“multi threads of control”).

Die Abarbeitung der Anweisungen kann entweder

1. echt gleichzeitig (echt parallel) oder
 2. zeitlich verzahnt (quasi-parallel)
- geschehen.

Verteiltes System (“distributed system“):

Verschiedene Teile des Software-Systems werden auf verschiedenen Rechnern ausgeführt, die miteinander vernetzt sind.

1.2 Parallele Programme

Grundidee:

Komplexes System wird in Teile zerlegt, die nebeneinander (parallel) agieren.

Typische Phänomene:

- Prozess-Synchronisation
 - wechselseitiger Ausschluss
(z.B. Zugriff auf gemeinsame Betriebsmittel oder gemeinsame Variable)
 - Kooperation von Prozessen
(z.B. Erzeuger/Verbraucher)
 - synchrone/asynchrone Nachrichtenübertragung
(z.B. Client/Server)
- Nichtdeterministisches Verhalten
- Sicherheit und Lebendigkeit (“Safety & Liveness“)
(Lebendigkeit: z.B. Fairness, kein Deadlock)

Vorteile:

- Gewinn von Performanz
- Adäquate Kontrollstruktur von Programmen bei Anwendungen mit inhärenter Parallelität
(z.B. bei eingebetteten Echtzeitsystemen oder Steuerung von Fertigungsprozessen)
- Besserer Durchsatz (z.B. bei Benutzerinteraktionen)

Probleme:

- Komplexität paralleler Systeme
- Abläufe paralleler Aktivitäten häufig schwer zu durchschauen

Konsequenz:

Fehlerhafte Programme

Benötigt:

Wohlfundierte Methoden und Techniken zum *Entwurf* und zur *Implementierung* paralleler Programme.

Wir verwenden dazu:

- im Entwurf:
 - *Modelle*, d.h. vereinfachte und abstrakte Darstellungen der parallelen Prozesse
 - *Beschreibungstechniken*:
 - graphisch: endliche Zustandsmaschinen
 - textuell: Sprache FSP (“finite state processes”)
 - *Techniken* zur Analyse und zum Nachweis von Sicherheits- und Lebendigkeitseigenschaften (Model Checking)
- in der Implementierung:
 - UML: Modellierung der Implementierung
 - Java: Codierung durch Java Threads, `synchronized methods`, `wait`, `notify`.

Der Übergang

Entwurfsmodell \rightsquigarrow Java-Programm

wird systematisch durchgeführt.