Modeling Concurrent and Probabilistic Systems

Lecture 1: Introduction

Joost-Pieter Katoen   Thomas Noll

Software Modeling and Verification Group
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/mcps09/

Summer Semester 2009

Outline

1. Preliminaries
2. Introduction
3. Syntax of CCS

People

	1st part: CCS	2nd part: Probabilistic Models
Lectures	Thomas Noll `<noll>`	Joost-Pieter Katoen `<katoen>`
Exercises	Daniel Klink `<klink>`	Tingting Han `<tingting.han>`
Assistant	Falak Sher `<chfalak@gmail.com>`	

(add "`@cs.rwth-aachen.de`" to e-mail addresses)

Target Audience

- Diploma programme (Informatik)
  - Theoretische Informatik
  - Vertiefungsfach Formale Methoden, Programmiersprachen und Softwarevalidierung
- Master programme (Software Systems Engineering)
  - Theoretical CS
  - Specialization Formal Methods, Programming Languages and Software Validation
- In general:
  - interest in formal models for software systems (in particular: *reactive* systems)
  - application of mathematical reasoning methods
- Expected: basic knowledge in
  - formal languages and automata theory
  - mathematical logic
  - probability theory

Organization

- Schedule:
  - Lecture Thu 15:00–16:30 AH 5 (starting April 23)
  - Lecture Fri 13:00–14:30 (?) AH 2 (starting April 24)
  - Exercise class Wed 13:30–15:00 AH 3 (starting May 6)
- see web page for complete list of dates
- 1st assignment sheet: Wed April 29 on web
- Work on assignments in groups of three
- Examination (8 ECTS credit points):
  written [July] or oral [July–Oct.]
  (depending on number of candidates)
- Admission requires at least 50% of the points in the exercises
- Solutions to exercises and exam in English or German

Outline

Motivation

Goal

Describing and analyzing the behavior of concurrent and/or probabilistic systems

Motivation

- Supporting the design phase
  - "Programming Concurrent Systems"
  - synchronization, scheduling, semaphores, ...
- Verifying functional correctness properties
  - "Model Checking"
  - validation of mutual exclusion, fairness, no deadlocks, ...
- Analyzing performance and dependability properties
  - "Performance Modeling"
  - queue throughput, response time in real-time systems, ...

Concurrency and Interaction I

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0123$$
$$13 \qquad\qquad 2$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1, or 3
- If exclusive access to shared memory and atomic execution of assignments guaranteed
  $\Longrightarrow$ only possible outcome: 3

Concurrency and Interaction II

The problem arises due to the combination of

- concurrency and
- interaction (here: via shared memory)

Conclusion

When modelling concurrent systems, the precise description of the mechanisms of both concurrency and interaction is crucially important.

Reactive Systems I

- Thus: "classical" model for sequential systems

$$\text{System} : \text{Input} \rightarrow \text{Output}$$

  (transformational systems) is not adequate
- Missing: aspect of interaction
- Rather: reactive systems which interact with environment and among themselves
- Main interest: not terminating computations but infinite behavior (system maintains ongoing interaction with environment)
- Examples:
  - operating systems
  - embedded systems controlling mechanical or electrical devices (planes, cars, home appliances, ...)
  - power plants, production lines, ...

Reactive Systems II

Observation: reactive systems often safety critical

$\Longrightarrow$ correct behavior has to be ensured

- Safety properties: "Nothing bad is going to happen."
  E.g., "at most one process in the critical section"
- Liveness properties: "Eventually something good will happen."
  E.g., "the server will finally answer"
- Fairness properties: "No component will starve to death."
  E.g., "any process requiring entry to the critical section will eventually be admitted"

Our approach I

The formal verification of such properties requires a mathematical model of the underlying system. Here we use the following approach:

- interaction is interpreted by explicit, synchronous communication and
- concurrency is modelled by interleaving, i.e., the (communication) actions of concurrent processes are merged:

$$(a;b) \parallel (x;y) \qquad \text{corresponds to} \qquad \begin{matrix} a \\ b \\ x \\ y \end{matrix} \quad \text{or} \quad \begin{matrix} a \\ x \\ b \\ y \end{matrix} \quad \text{or} \quad \begin{matrix} x \\ a \\ b \\ y \end{matrix} \quad \text{or} \ \ldots$$

$\Longrightarrow$ reduction of concurrency to nondeterminism
(cf. multitasking on sequential computers)

Possible alternatives:

- interaction via shared memory/asynchronous message passing/...
- concurrency via true parallelism (Petri Nets)
- later: probabilistic aspects [Katoen]

Our approach II

"Primary meaning" of a system: potential of communication
i.e., the set of possible communication sequences

In particular:

- I/O modelled as communication with environment
- storage access modelled as communication with a "storage process"

Overview of the Course

- 1st part of course (CCS):
  2. Calculus of Communicating Systems (CCS)
     (syntax, labeled transition systems, transition rules)
  3. Equivalence of CCS Processes
     (trace equivalence, strong/weak bisimulation, observation congruence, axiomatizability of equivalences)
  4. Case Study: Alternating-Bit Protocol
     (modeling channels/sender/receiver, correctness, extensions)
- 2nd part of course (Probabilistic Models):
  5. Stochastic processes
     (Markov chains and decision processes)
  6. Probabilistic (bi)simulation
     (strong bisimulation/simulation, simulation equivalence)
  7. Probabilistic process algebra
     (probabilistic transition systems, operators, axiomatizability of probabilistic bisimulation)
  8. Further Issues
     (nondeterminism, continuous time, Markovian process algebra)

Literature

(also see the collection ["Handapparat Probabilistic Models for Concurrency / PMC"] at the CS Library)

- 1st part of course (CCS):
  - R. Milner: *Communication and Concurrency*
    Prentice-Hall, 1989
  - R. Milner: *Communicating and Mobile Systems: the π-calculus*
    Cambridge University Press, 1999
  - J.A. Bergstra, A. Ponse, S.A. Smolka: *Handbook of Process Algebra*
    Elsevier, 2001
- 2nd part of course (Probabilistic Models):
  - H.C. Tijms: *A first course in stochastic models*
    Wiley, 2003
  - J. Hillston: *A Compositional Approach to Performance Modelling*
    Cambridge University Press, 1996
  - H. Hermanns: *Interactive Markov Chains: The Quest for Quantified Quality*
    LNCS 2428, Springer, 2002
  - E. Brinksma, H. Hermanns, J.-P. Katoen: *Lectures on Formal Methods and Performance Analysis*, LNCS 2090, Springer, 2001

Outline

History of CCS

- Robin Milner: *A Calculus of Communicating Systems*
  LNCS 92, Springer, 1980
- Robin Milner: *Communication and Concurrency*
  Prentice-Hall, 1989
- Robin Milner: *Communicating and Mobile Systems: the π-calculus*
  Cambridge University Press, 1999

Approach: describing concurrency on a simple and abstract level, using only a few basic primitives

- no explicit storage (variables)
- no explicit representation of values (numbers, Booleans, ...)

$\Longrightarrow$ concurrent system reduced to communication potential

Syntax of CCS I

Definition 1.2 (Syntax of CCS)

- Let N be a set of (action) names.
- $\overline{N} := \{\overline{a} \mid a \in N\}$ denotes the set of co-names.
- $Act := N \cup \overline{N} \cup \{\tau\}$ is the set of actions where τ denotes the silent (or: unobservable) action.
- Let Pid be a set of process identifiers.
- The set Prc of process expressions is defined by the following syntax:
$$\begin{array}{rcll} P & ::= & \mathsf{nil} & \text{(inaction)} \\ & \mid & \alpha.P & \text{(prefixing)} \\ & \mid & P_1 + P_2 & \text{(choice)} \\ & \mid & P_1 \parallel P_2 & \text{(parallel composition)} \\ & \mid & \mathsf{new}\, a\, P & \text{(restriction)} \\ & \mid & A(a_1, \ldots, a_n) & \text{(process call)} \end{array}$$
where $\alpha \in Act$, $a, a_i \in N$, and $A \in Pid$.

Syntax of CCS II

Definition 1.2 (continued)

- A (recursive) process definition is an equation system of the form

$$(A_i(a_{i1}, \ldots, a_{in_i}) = P_i \mid 1 \leq i \leq k)$$

where $k \geq 1$, $A_i \in Pid$ (pairwise different), $a_{ij} \in N$ ($a_{i1}, \ldots, a_{in_i}$ pairwise different), and $P_i \in Prc$ (with process identifiers from $\{A_1, \ldots, A_k\}$).

Meaning of CCS Constructs

- nil is an inactive process that can do nothing.
- $\alpha.P$ can execute α and then behaves as P.
- An action $a \in N$ ($\overline{a} \in \overline{N}$) is interpreted as an input (output, resp.) operation. Both are complementary: if executed in parallel (i.e., in $P_1 \parallel P_2$), they are merged into a τ-action.
- $P_1 + P_2$ represents the non-deterministic choice between P_1 and P_2.
- $P_1 \parallel P_2$ denotes the concurrent execution of P_1 and P_2, involving interleaving or communication.
- The restriction $\mathsf{new}\, a\, P$ declares a as a local name which is only known in P.
- The behavior of a process call $A(a_1, \ldots, a_n)$ is defined by the right-hand side of the corresponding equation where $a_1, \ldots, a_n$ replace the formal name parameters.

CCS Examples

Example 1.3

1. One-place buffer
2. Two-place buffer
3. Parallel specification of two-place buffer

(on the board)

Notational Conventions

- $\overline{\overline{a}}$ means a
- $P_1 + \ldots + P_n$ ($n \in \mathbb{N}$) sometimes written as $\sum_{i=1}^{n} P_i$ where $\sum_{i=1}^{0} P_i := \mathsf{nil}$
- ".nil" can be omitted: $a.b$ means $a.b.\mathsf{nil}$
- $\mathsf{new}\, a, b\, P$ means $\mathsf{new}\, a\, \mathsf{new}\, b\, P$
- $A(a_1, \ldots, a_n)$ sometimes written as $A(\vec{a})$, $A()$ as A
- prefixing and restriction binds stronger than composition, composition binds stronger than choice:

$$\mathsf{new}\, a\, P + b.Q \parallel R \quad \text{means} \quad (\mathsf{new}\, a\, P) + ((b.Q) \parallel R)$$