Process Algebra (2IMF10)

Introduction and motivation

Bas Luttik

MF 6.072
s.p.luttik@tue.nl
http://www.win.tue.nl/~luttik

Lecture 1

Semantics of computing

Classical view

A computer program transforms an input into an output.

According to this view, the *denotational semantics* of a program P is a partial function

$$[\![P]\!] : States \rightharpoonup States \ .$$

Why is it a partial function?

Traditionally, programs are supposed to terminate; non-termination was considered bad!

Is this all we need?

Reactive systems

What about

- Vending machines?
- Operating systems?
- Communication protocols?
- Plant controllers?
- Mobile phones?

Reactive systems

Reactive systems

Characterisation

A reactive system is a system that computes by reacting to stimuli from its environment.

Key issues:

- parallelism
- communication and interaction
- non-termination is often a good thing

Analysis of reactive systems

Typical questions

- How can we develop (design) a system that works?
- How can we analyse (verify) the design?

The course *System Validation* addresses these questions from a pragmatic point of view.

Why would we analyse in the first place?

Fact of life

Even very short parallel programs may be very hard to understand and may exhibit unexpected behaviour.

Concurrency theory

What is it?

- an active research field in theoretical computer science
- that studies formalisms (Petri nets, process calculi, ...) suitable for modelling and analysing system behaviour;
- it is mainly concerned with **reactive systems**.

What's the use?

- it helps to understand phenomena occurring in parallel and distributed systems;
- it facilitates reasoning about such systems;
- it is used to provide solid semantics of specification languages (e.g., mCRL2).

Process algebra ...

- branch of concurrency theory
- focusses on operations for *composing* behaviour

It (originally) aimed at reasoning about processes by *calculation*.

Can you think of sensible operations on behaviour?

Process algebra ...

Basic principle

1. Define a collection of atomic processes (modelling the simplest behaviour).
2. Define new composition operators (for building more complex behaviour from simpler behaviours).

Example

1. atomic instruction: assignment (e.g., $x := 2$ and $x := x + 2$)
2. operators:
   - sequential composition ($P_1 \cdot P_2$)
   - parallel composition ($P_1 \| P_2$)

Process algebra ...

What is the value of x after executing the process

$$(x := 1 \| x := 2) \cdot x := x + 2 \cdot (x := x - 1 \| x := x + 5)$$

Process algebra ...

Process algebra is in particular concerned with the semantics and expressiveness of constructs used to formally specify behaviour.

We shall consider:

- Various notions of composition (alternative composition, sequential composition, parallel composition, etc.)
- Expressiveness (esp. in the context of recursion)
- Various notions of behavioural equivalence
- Axiom systems and their quality (soundness/completeness)
- Abstraction
- ...

Course organisation

For now, information can be found at

`http://www.win.tue.nl/~luttik/Courses/PA`

Should we try to move this to Canvas?

Course organisation: contents

Tentative schedule

- week 1: algebra, math. repr. of behaviour
- week 2: basic process algebra
- week 3: extension (example: sequential composition)
- week 4: recursion
- week 5: expressiveness and definability
- week 6: parallelism, interaction
- week 7: abstraction
- week 8: advanced topic (T.B.D.)

Course organisation: material

We will treat a significant part of the following book:

J. C. M. Baeten, T. Basten and M. A. Reniers.
Process Algebra: Equational Theories of Communicating Processes.
Cambridge University Press, 2010.

Full text can be downloaded via `library.tue.nl`.

Furthermore, we shall make use of slides, several research articles.

Course organisation: assessment

Assignments

Three homework assignments (consisting of sets of exercises):

1. deadline: Sunday May 14, 2017;
2. deadline: Sunday May 28, 2017;
3. deadline: Sunday June 11, 2017.

Together, the assignments count for 30% of your grade.

Final exam

1. Tuesday June 27, 2017 (13:30-16:30)
2. Tuesday August 15, 2017 (13:30–16:30)

The final exam counts for 70% of your grade.

You may bring paper copies of slides and the book (no annotations, no notes!)

Course organisation: lectures

- two lectures a week
- be active
- give feedback to the lecturer
- do exercises (see schedule on website)

Prerequisites

This will be a *Theoretical Computer Science* course.

You should

- be familiar with (formal) logical reasoning (i.e., you should be able to give a mathematical proof, i.p. using various kinds of induction)
- be familiar with (naive) set theory
- preferably have had some prior exposure to automata and process theory