Semantics and Verification

Lecture 1

2 February 2010

About this Course

Focus of the Course

- Study of mathematical models for the formal description and analysis of programs
- Particular focus on parallel and reactive systems
- Verification tools: How to use them, and how they work "under the hood"

Overview of the Course

- Transition systems and CCS
- Strong and weak bisimilarity, bisimulation games
- Hennessy-Milner logic and bisimulation
- Tarski's fixed-point theorem
- Hennessy-Milner logic with recursively defined formulae
- Tined CCS
- Timed automata and their semantics
- Binary decision diagrams and their use in verification

- Two mini projects

Mini Projects

- Modeling and verification of a communication protocol in CWB-NC
- Modeling and verification of a real-time algorithm in UPPAAL

- Take a real-world problem and model it as a formal system
- Formulate interesting properties of the problem in a formal language
- Use the verification tool to check whether the system satisfies the properties

$\Rightarrow$ Pensum dispensation

Aims of the Course

Present a general theory of reactive systems and its applications

- Design
- Specification
- Verification (possibly automatic and compositional)

1. Give the students practice in modelling parallel systems in a formal framework
2. Give the students skills in analyzing behaviours of reactive systems
3. Introduce algorithms and tools based on the modelling formalisms

Lectures

- Ask questions
- Lot of material not covered in slides, only on blackboard

$\Rightarrow$ Take your own notes

- Read the required material
  - before the lecture, fast
  - (so you know what we'll talk about)
  - and again after the lecture, thoroughly
  - (so you're sure you understand)

Tutorials

- Always before each lecture
- Supervised peer learning
- Work in groups of 2 or 3 people
- Print out the exercises, bring literature and your notes
- Ask us for help and feedback
- Starred exercises $\Rightarrow$ exam

Classical View

Characterization of a Classical Program

A program transforms an input into an output.

- Denotational semantics:
  meaning of a program is a partial function

$$\mathit{states} \hookrightarrow \mathit{states} \qquad ; \qquad \mathit{input} \mapsto \mathit{output}$$

- Nontermination is bad!
- In case of termination, the result is unique.

Is this all we need?

Reactive Systems

What about:

- Operating systems?
- Communication protocols?
- Control programs?
- Mobile phones?
- Vending machines?

Reactive Systems

Characterization of a Reactive System

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

Key Issues:

- communication and interaction
- parallelism

- Nontermination is good!
- The result (if any) does not have to be unique.

Analysis of Reactive Systems

Questions

- How can we develop (design) a system that "works"?
- How do we analyze (verify) such a system?

Fact of Life

Even short parallel programs may be hard to analyze.

The Need for a Theory

Conclusion

We need formal/systematic methods (tools), otherwise ...

- Intel's Pentium-II bug in floating-point division unit
- Ariane-5 crash due to a conversion of 64-bit real to 16-bit integer
- Mars Pathfinder
- ...

before

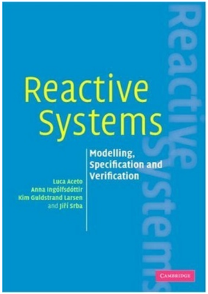
after

Exam

- Oral, 20 min
- Randomly choose 1 out of 9 examination subjects
- 20 min preparation time
- Internal censor; pass / fail

- Subjects are known before-hand $\Rightarrow$ prepare!
- Starred exercises
- Each mini project makes one examination subject
- Successfully solve mini project $\Rightarrow$ pensum dispensation!

Course book: Reactive Systems: Modelling, Specification and Verification. Aceto, Ingólfsdóttír, Larsen, Srba. Cambridge University Press, 2007

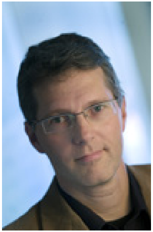


http://rsbook.cs.aau.dk

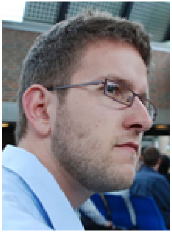
- plus a number of articles, tutorials etc.

People

Uli Fahrenberg
lecturer
uli@cs.aau.dk



Kim G. Larsen
lecturer
kgl@cs.aau.dk



Mikkel L. Pedersen
part-time TA
mikkelp@cs.aau.dk

Hints

- Check the course web page frequently

  https://intranet.cs.aau.dk/education/courses/2010/sv/
- Give us feedback: comments, suggestions, criticism
- Or use anonymous feedback form at http://dw3.cs.aau.dk/~uli/sv10/form.html
- Attend the lectures
- Take your own notes
- Participate in the tutorials

Formal Models for Reactive Systems

Classical vs. Reactive Computing

	Classical	Reactive
interaction	no	yes
nontermination	undesirable	often desirable
unique result	yes	no
semantics	denotational *states* $\hookrightarrow$ *states*	operational

How to Model Reactive Systems

Question

What is the most abstract view of a reactive system (process)?

Answer

A process performs an action and becomes another process.

Labelled Transition Systems

Definition

A labelled transition system (LTS) is a triple $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ where

- $Proc$ is a set of states (or processes),
- Act is a set of labels (or actions), and
- for every $a \in Act$, $\xrightarrow{a} \subseteq Proc \times Proc$ is a binary relation on states called the transition relation.

We will use infix notation: $s \xrightarrow{a} s'$ means that $(s, s') \in \xrightarrow{a}$

Sometimes we distinguish the initial (or start) state.

Sequencing, Nondeterminism and Parallelism

LTS explicitly focus on interaction.

LTS can also describe:

- sequencing $(a; b)$
- choice (nondeterminism) $(a + b)$
- limited notion of parallelism (by using interleaving) $(a \mid b)$

Binary Relations

Definition

A binary relation R on a set A is a subset of $A \times A$.

$$R \subseteq A \times A$$

Sometimes we write $x\,R\,y$ instead of $(x, y) \in R$.

Properties

- R is called reflexive if $(x, x) \in R$ for all $x \in A$
- R is called symmetric if for all $x, y \in A$, $(x, y) \in R$ implies that $(y, x) \in R$
- R is called transitive if for all $x, y, z \in A$, $(x, y) \in R$ and $(y, z) \in R$ imply that $(x, z) \in R$

Reflexive Closure

Let R and R' be binary relations on a set A.

Definition

R' is called a reflexive closure of R if

1. $R \subseteq R'$,
2. R' is reflexive, and
3. R' is minimal with respect to the two conditions above:
   For any binary relation R'' on A:
   if $R \subseteq R''$ and R'' is reflexive, then $R' \subseteq R''$.

Fact: Any binary relation has a unique reflexive closure.

Symmetric Closure

Let R and R' be binary relations on a set A.

Definition

R' is called a symmetric closure of R if

1. $R \subseteq R'$,
2. R' is symmetric, and
3. R' is minimal with respect to the two conditions above: For any binary relation R'' on A: if $R \subseteq R''$ and R'' is symmetric, then $R' \subseteq R''$.

Fact: Any binary relation has a unique symmetric closure.

Transitive Closure

Let R and R' be binary relations on a set A.

Definition

R' is called a transitive closure of R if

1. $R \subseteq R'$,
2. R' is transitive, and
3. R' is minimal with respect to the two conditions above:
   For any binary relation R'' on A:
   if $R \subseteq R''$ and R'' is transitive, then $R' \subseteq R''$.

Fact: Any binary relation has a unique transitive closure.

Let $(Proc, Act, \{\xrightarrow{a} \mid a \in Act\})$ be an LTS.

- we extend $\xrightarrow{a}$ to the elements of Act^*
- $\longrightarrow = \bigcup_{a \in Act} \xrightarrow{a}$
- $\longrightarrow^*$ is the reflexive and transitive closure of $\longrightarrow$
- $s \xrightarrow{a}$ and $s \stackrel{a}{\not\longrightarrow}$
- reachable states

How to Describe LTS?

Syntax		Semantics
unknown entity	$\longrightarrow$	known entity
programming language	$\longrightarrow$	what (denotational) or how (operational) it computes
CCS	$\longrightarrow$	Labelled Transition Systems

Calculus of Communicating Systems

CCS

Process algebra called "Calculus of Communicating Systems".

Insight of Robin Milner (1989)

Concurrent (parallel) processes have an algebraic structure.

$$\boxed{P_1}\ op\ \boxed{P_2} \Rightarrow \boxed{P_1\ op\ P_2}$$

Process Algebra

Basic Principle

1. Define a few atomic processes (modelling the simplest process behaviour).
2. Define compositionally new operations (building more complex process behaviour from simple ones).

Example

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

   Now e.g. (x:=1 | x:=2); x:=x+2; (x:=x-1 | x:=x+5) is a process.

CCS Basics (Sequential Fragment)

- *Nil* (or 0) process (the only atomic process)
- action prefixing $a.P$
- names and recursive definitions via $\stackrel{\text{def}}{=}$
- nondeterministic choice $+$

This is Enough to Describe Sequential Processes

Any finite LTS can be (up to isomorphism) described by using the operations above.