Reactive Systems: Modelling, Specification and Verification

EWSCS'07

Anna Ingolfsdottir
Reykjvik University, Iceland

Focus of the Course

- Study of mathematical models for the formal description and analysis of programs.
- Study of formal languages for the specification of program behaviour.
- Particular focus on parallel and reactive systems.
- Verification tools and implementation techniques underlying them.

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.

Aims of the Course

Present a general theory of reactive systems and its applications.
The theory supports:

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

Aims

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.

Aims of the Course

Present a general theory of reactive systems and its applications. The theory supports:

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

Aims

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.

Aims of the Course

Present a general theory of reactive systems and its applications.
The theory supports:

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

Aims

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.

Aims of the Course

Present a general theory of reactive systems and its applications.
The theory supports:

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

Aims

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.

Classical View

Characterization of a Classical Program

Program transforms an input into an output.

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

$$states \hookrightarrow states$$

- 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

Reactive System = 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.

Reactive systems

Characterization of a Reactive System

Reactive System = 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
- ...

Classical vs. Reactive Computing

	Classical	Reactive/Parallel
interaction	no	yes
nontermination	undesirable	often desirable
unique result	yes	no
semantics	$states \hookrightarrow states$	?

How to Model Reactive Systems

Question

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

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
- $\xrightarrow{a} \subseteq Proc \times Proc$ is a binary relation on states called the transition relation, for each $a \in Act$.

We will use the infix notation $s \xrightarrow{a} s'$ meaning that $(s, s') \in \xrightarrow{a}$.

Sometimes we distinguish an initial (or start) state.

Keyword: Interaction!

LTSes describe process behaviour, and explicitly focus on interaction.

The Motto (after Tony Hoare and Robin Milner)

Everything is (or can be viewed as) a process!

Buffers, shared memory, Linda tuple spaces, senders, receivers, . . . are all agents/processes.

Labelled Transition Systems – Notation

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 $\longrightarrow$. (Do you know what this means?)
- $s \xrightarrow{a}$ and $s \not\xrightarrow{a}$.
- Reachable states.

How to Describe LTSes?

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

CCS (Milner 1980)

How to Describe LTSes?

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

CCS (Milner 1980)

How to Describe LTSes?

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

CCS (Milner 1980)

How to Describe LTSes?

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

CCS (Milner 1980)

Calculus of Communicating Systems

CCS

Process algebra called "Calculus of Communicating Systems".

Insight of Robin Milner (1980, developed from earlier work)

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 new composition operations (building more complex process behaviour from simpler 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 ($P_1 \parallel P_2$)

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

Process Algebra

Basic Principle

1. Define a few atomic processes (modelling the simplest process behaviour).
2. Define new composition operations (building more complex process behaviour from simpler 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 ($P_1 \parallel P_2$)

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

A CCS Process: Black-Box View

What is a CCS Process to its Environment?

A CCS process is a computing agent that may communicate with its environment via its interface.
Interface = Collection of communication ports/channels, together with an indication of whether they are used for input or output.

Example: A Computer Scientist

Process interface: coffee (input port)
$\overline{\text{coin}}$, $\overline{\text{pub}}$ (output ports)

Question: How do we describe the behaviour of the "black-box"?

A CCS Process: Black-Box View

What is a CCS Process to its Environment?

A CCS process is a computing agent that may communicate with its environment via its interface.
Interface = Collection of communication ports/channels, together with an indication of whether they are used for input or output.

Example: A Computer Scientist

Process interface: coffee (input port)
$\overline{\text{coin}}$, $\overline{\text{pub}}$ (output ports)

Question: How do we describe the behaviour of the "black-box"?

CCS Basics (Sequential Fragment)

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

This is Enough to Describe Sequential Processes

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

CCS Basics (Sequential Fragment)

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

This is Enough to Describe Sequential Processes

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