Semantics and Verification 2006

Lecture 15

- round-up of the course
- information about the exam
- selection of star exercises

Reactive systems

Characterization of a Reactive System

Reactive System is a system that computes by reacting to stimuli from its environment.

Key Issues:

- parallelism
- communication and interaction

Nontermination is good!

The result (if any) **does not have to be unique**.

Classical vs. Reactive Computing

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

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 Algebras

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 ($P_1 \mid P_2$)

Usually given by **abstract syntax**:

$$P, P_1, P_2 \ ::= \ x := e \ \mid \ P_1; P_2 \ \mid \ P_1 | P_2$$

where x ranges over variables and e over arithmetical expressions.

Syntax of CCS

Process expressions:

$P :=$	K	$\mid$	process constants ($K \in \mathcal{K}$)
	$\alpha.P$	$\mid$	prefixing ($\alpha \in Act$)
	$\sum_{i \in I} P_i$	$\mid$	summation (I is an arbitrary index set)
	$P_1 \mid P_2$	$\mid$	parallel composition
	$P \smallsetminus L$	$\mid$	restriction ($L \subseteq \mathcal{A}$)
	$P[f]$	$\mid$	relabelling ($f : Act \rightarrow Act$) such that

- $f(\tau) = \tau$
- $f(\bar{a}) = \overline{f(a)}$

$P_1 + P_2 = \sum_{i \in \{1,2\}} P_i$ $\qquad$ $Nil = 0 = \sum_{i \in \emptyset} P_i$

CCS program

A collection of **defining equations** of the form $K \stackrel{\text{def}}{=} P$ where $K \in \mathcal{K}$ is a process constant and P is a process expression.

Semantics of CCS — SOS rules ($\alpha \in Act$, $a \in \mathcal{L}$)

$$\text{ACT}\quad \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{SUM}_j\quad \frac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i\in I} P_i \xrightarrow{\alpha} P'_j}\ j \in I$$

$$\text{COM1}\quad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \text{COM2}\quad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\text{COM3}\quad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\text{RES}\quad \frac{P \xrightarrow{\alpha} P'}{P \smallsetminus L \xrightarrow{\alpha} P' \smallsetminus L}\ \alpha, \overline{\alpha} \notin L \qquad \text{REL}\quad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

$$\text{CON}\quad \frac{P \xrightarrow{\alpha} P'}{K \xrightarrow{\alpha} P'}\ K \stackrel{\text{def}}{=} P$$

Verification Approaches

Let *Impl* be an implementation of a system (e.g. in CCS syntax).

Equivalence Checking Approach

$$Impl \equiv Spec$$

- *Spec* is a **full specification** of the intended behaviour
- Example: $s \sim t$ or $s \approx t$

Model Checking Approach

$$Impl \models Property$$

- *Property* is a **partial specification** of the intended behaviour
- Example: $s \models \langle a\rangle([b]ff \wedge \langle a\rangle tt)$

Relationship between Equivalence and Model Checking

- Equivalence checking and model checking are **complementary** approaches.
- They are strongly connected, however.

Theorem (Hennessy-Milner)

Let us consider an image-finite LTS. Then

$$p \sim q$$

if and only if

for every HM formula F (even with recursion): $(p \models F \iff q \models F)$.

Introducing Time Features

In many applications, we would like to explicitly model **real-time** in our models.

Timed (labelled) transition system

Timed LTS is an ordinary LTS where actions are of the form $Act = L \cup \mathbb{R}^{\geq 0}$.

- $s \xrightarrow{a} s'$ for $a \in L$ are discrete transitions
- $s \xrightarrow{d} s'$ for $d \in \mathbb{R}^{\geq 0}$ are time-elapsing (delay) transitions

Timed and Untimed Bisimilarity

Let s and t be two states in timed LTS.

Timed Bisimilarity (= Strong Bisimilarity)

We say that s and t are **timed bisimilar** iff $s \sim t$.

Remark: all transitions are considered as **visible** transitions.

Untimed Bisimilarity

We say that s and t are **untimed bisimilar** iff $s \sim t$ in a modified transition system where every transition of the form $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$ is replaced by a transition $\xrightarrow{\epsilon}$ for a new (single) action ϵ.

Remark:

- $\xrightarrow{a}$ for $a \in L$ are treated as visible transitions, while
- $\xrightarrow{d}$ for $d \in \mathbb{R}^{\geq 0}$ all look the same (action ϵ).

Timed CCS — a Way to Define Timed LTS

Syntax of CCS with Time Delays

All CCS operators +

if P is a process then $\epsilon(d).P$ is also a process for any nonnegative real number d

Semantics of CCS with Time Delays

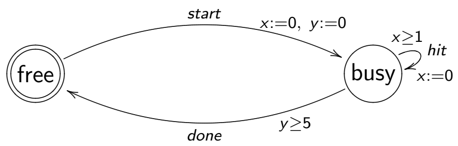
By means of SOS rules

- standard CCS rules
- SOS rules for time delays (maximal progress assumption)

we describe for a given TCCS expression what is the corresponding timed transition system.

Timed Automata — a Way to Define Timed LTS

- Nondeterministic finite-state automata with additional time features (**clocks**).
- Clocks can be tested against constants or compared to each other (pairwise).
- Executing a transition can reset selected clocks.



Region Graph — a Verification Technique for TA

We introduce an equivalence on clock valuations ($v \equiv v'$) with **finitely many** equivalence classes.

state (ℓ, v) $\rightsquigarrow$ **symbolic state** $(\ell, [v])$

Region Graph Preserves Untimed Bisimilarity

For every location ℓ and any two valuations v and v' from the same symbolic state ($v \equiv v'$) it holds that (ℓ, v) and (ℓ, v') are untimed bisimilar.

Reduction of Timed Automata Reachability to Region Graphs

$(\ell_0, v_0) \longrightarrow^* (\ell, v)$ in a timed automaton if and only if

$(\ell_0, [v_0]) \Longrightarrow^* (\ell, [v])$ in its (finite) region graph.

Compact Representation of State-Spaces in the Memory

Boolean Functions (where $\mathbb{B} = \{0, 1\}$)

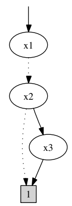
$$f : \mathbb{B}^n \to \mathbb{B}$$

Boolean Expressions

$$t, t_1, t_2 ::= 0 \mid 1 \mid x \mid \neg t \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid t_1 \Rightarrow t_2 \mid t_1 \Leftrightarrow t_2$$

Boolean expression:

$\neg x_1 \wedge (x_2 \Rightarrow (x_1 \vee x_3))$



Reduced and Ordered Binary Decision Diagram (ROBDD)

Logical operations on ROBDDs can be done efficiently!

The End

The course is over now!

Thanks

Thank you for proof-reading the book!

Information about the Exam

- Oral exam with preparation time, passed/failed.
- Preparation time (20 minutes) for solving a randomly selected star exercise.
- Examination time (20 minutes):
  - presentation of the star exercise (necessary condition for passing)
  - presentation of your randomly selected exam question
  - answering questions
  - evaluation
- 9 exam questions (with possible pensum dispensation).
- For a detailed summary of the reading material check the lectures plan.

Exam Questions

1. Transition systems and CCS.
2. Strong and weak bisimilarity, bisimulation games.
3. Hennessy-Milner logic and bisimulation.
4. Tarski's fixed-point theorem and Hennessy-Milner logic with one recursive formulae.
5. Alternating bit protocol and its modelling and verification using CWB. (**Possible pensum dispensation.**)
6. Timed CCS and bisimilarity.
7. Timed automata.
8. Gossiping girls problem and its modelling and verification using UPPAAL. (**Possible pensum dispensation.**)
9. Binary decision diagrams and their applications.

Further details are on the web-page. Check whether you are on the list of students with pensum dispensation!

How to Prepare for the Exam?

- Read the recommended material.
- Try to understand all topics equally well (remember you pick up two random topics out of 7).
- Go through all tutorial exercises and try to solve them. (Make sure that you can solve all star exercises fast!)
- Go through the slides to see whether you didn't miss anything.
- Make a summary for each question on a few A4 papers (you can take them at exam).
- Prepare a strategy how to present each question.

Further Tips

- It does not matter if you make a small error in a star exercise (as long as you understand what you are doing).
- Present a solution to the star exercise quickly (max 5 minutes).
- Start your presentation by writing a road-map (max 4 items).
- Plan your presentation to take about 10 minutes:
  - give a good overview
  - do not start with technical details
  - use the blackboard
  - use examples (be creative)
  - say only things that you know are correct
  - be ready to answer supplementary questions
  - tell a story (covering a sufficient part of the exam question)

Examples of Star Exercises — CCS

- By using SOS rules for CCS prove the existence of the following transition (assume that $A \stackrel{\text{def}}{=} a.A$):

$$((A \mid \overline{a}.Nil) + A) \smallsetminus \{a\} \stackrel{\tau}{\longrightarrow} (A \mid Nil) \smallsetminus \{a\}$$

- Draw the LTS generated by the following CCS expression:

$$(\overline{a}.Nil \mid a.Nil) + b.Nil$$

Examples of Star Exercises — Bisimilarity

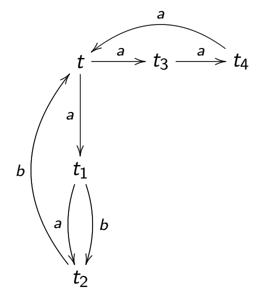
Determine whether the following two CCS expressions

$$a.(b.Nil + c.Nil) \quad \text{and} \quad a.(b.Nil + \tau.c.Nil)$$

are:

- strongly bisimilar?
- weakly bisimilar?

Examples of Star Exercises — HML



Determine whether

- $t \models [a](\langle b \rangle tt \vee [a][b]ff)$
- $t \models X$ where $X \stackrel{\max}{=} \langle a \rangle tt \wedge [Act]X$

Find a distinguishing formulae for the CCS expressions:

$$a.a.Nil + a.b.Nil \quad \text{and} \quad a.(a.Nil + b.Nil).$$

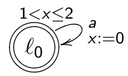
Examples of Star Exercises — TCCS

Using the SOS rules for TCCS prove that

$$\epsilon(5).(\epsilon(3).Nil + b.Nil) \xrightarrow{7} \epsilon(1).Nil + b.Nil$$

Examples of Star Exercises — TA

Draw a region graph of the following timed automaton:



Examples of Star Exercises — ROBDD

Construct ROBDD for the following boolean expression:

$$x_1 \wedge (\neg x_2 \vee x_1 \vee x_2) \wedge x_3$$

such that $x_1 < x_2 < x_3$.