Introduction to Infinite-State Systems

Jiri Srba, BRICS Aalborg

30.11 – 1.12 2005
PhD Course at FIRST Graduate School, IT University

Overview of the Course

- Mathematical models for the formal description and analysis of programs with unbounded state spaces.
- Decidability and complexity issues.
- Applicability of the results.

Aims of the Course

Present a general theory of reactive systems with infinite-state spaces and their applications.

- What does "equivalent" mean for reactive systems?
- Description of infinite-state systems.
- Selected techniques and results.

1. Give the participants an overview of the area.
2. Show in detail a few key techniques with wider applicability.
3. Motivate the participants to have a closer look at topics of interest.
(participation = 1.5 ECTS; participation+essay = 3.5 ECTS)

Literature

- On-line literature at
`http://www.cs.aau.dk/~srba/courses/PhD-05/first.html`
- Take your notes and participate actively.

Overview of the Course Topics

- reactive systems, LTS, bisimilarity, games
- process rewrite systems — hierarchy of infinite-state systems
- decidability of $\sim$ for BPP — tableau technique
- complexity of $\sim$ for BPP — defender's choice technique
- reachability for PDA in P — automata-theoretical approach
- applicability to control-flow analysis
- undecidability of $\sim$ for PN — Minsky machine and reductions
- well quasi ordering — a universal decidability theorem
- visibly pushdown automata — a class with nice closure properties
- ...

Classical View on Computations

Characterization of a Classical Program

Program transforms an input into an output.

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

$$\textit{states} \hookrightarrow \textit{states}$$

- 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 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.

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)?

Answer

A process performs an action and becomes another process.

Labelled Transition System

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

Sometimes we distinguish the initial (or start) state.

Sequencing, Nondeterminism and Parallelism

LTS explicitly focuses on interaction.

LTS can also describe:

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

Our focus: LTS with infinitely (even uncountably) many states.

Verifying Correctness of Reactive Systems

Let *Impl* be an implementation of a system (given as an LTS).

Equivalence Checking Approach

$$Impl \equiv Spec$$

- $\equiv$ is an abstract equivalence, e.g. strong bisimilarity
- *Spec* is also an LTS
- *Spec* provides the full specification of the intended behaviour

Model Checking Approach

$$Impl \models Property$$

- $\models$ is the satisfaction relation
- *Property* is a particular feature, often expressed via a logic
- *Property* is a partial specification of the intended behaviour

Strong Bisimilarity

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

Strong Bisimulation

A binary relation $R \subseteq Proc \times Proc$ is a strong bisimulation iff whenever $(s, t) \in R$ then for each $a \in Act$:

- if $s \xrightarrow{a} s'$ then $t \xrightarrow{a} t'$ for some t' such that $(s', t') \in R$
- if $t \xrightarrow{a} t'$ then $s \xrightarrow{a} s'$ for some s' such that $(s', t') \in R$.

Strong Bisimilarity

Two processes $p_1, p_2 \in Proc$ are strongly bisimilar ($p_1 \sim p_2$) if and only if there exists a strong bisimulation R such that $(p_1, p_2) \in R$.

$$\sim \; = \; \cup\{R \mid R \text{ is a strong bisimulation}\}$$

Strong Bisimulation Game

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

We define a two-player game of an 'attacker' and a 'defender' starting from s and t.

- The game is played in rounds and configurations of the game are pairs of states from $Proc \times Proc$.
- In every round exactly one configuration is called current. Initially the configuration (s, t) is the current one.

Intuition

The defender wants the show that s and t are strongly bisimilar while the attacker aims to prove the opposite.

Rules of the Bisimulation Games

Game Rules

In each round the players change the current configuration as follows:

1. the attacker chooses one of the processes in the current configuration and makes an $\xrightarrow{a}$-move for some $a \in Act$, and
2. the defender must respond by making an $\xrightarrow{a}$-move in the other process under the same action a.

The newly reached pair of processes becomes the current configuration. The game then continues by another round.

Result of the Game

- If one player cannot move, the other player wins.
- If the game is infinite, the defender wins.

Game Characterization of Strong Bisimilarity

Theorem

- States s and t are strongly bisimilar if and only if the defender has a universal winning strategy starting from the configuration (s, t).
- States s and t are not strongly bisimilar if and only if the attacker has a universal winning strategy starting from the configuration (s, t).

Remark

Bisimulation game can be used to prove both bisimilarity and nonbisimilarity of two processes. It very often provides elegant arguments for the negative case.

Process Algebra - a Way to Describe Infinite-State Systems

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).