Test and Verification

Brian Nielsen

bnielsen@cs.aau.dk
## Preliminary Plan

<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>SW8</th>
<th>SP2</th>
<th>Lecture Date</th>
<th>Lecture Time</th>
<th>Exercise Room</th>
<th>Lecturer</th>
<th>Slides</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Feb 5</td>
<td>0</td>
<td>0</td>
<td>Feb 5</td>
<td>9:15-10:00</td>
<td>PC-Lab</td>
<td>BN</td>
<td>Introduction</td>
<td>Introduction</td>
</tr>
<tr>
<td>2.</td>
<td>Feb 7</td>
<td>0</td>
<td>0</td>
<td>Feb 7</td>
<td>9:15-10:00</td>
<td>PC-Lab</td>
<td>BN</td>
<td>Modelling in UPPAAL</td>
<td>Modelling in UPPAAL, Timed Automata</td>
</tr>
<tr>
<td>3.</td>
<td>Feb 12</td>
<td>0</td>
<td>0</td>
<td>Feb 12</td>
<td>9:15-10:00</td>
<td>PC-Lab</td>
<td>AD</td>
<td>Engine and Options</td>
<td>Verification Engine and Options of UPPAAL</td>
</tr>
<tr>
<td>4.</td>
<td>Feb 14</td>
<td>0</td>
<td>0</td>
<td>Feb 14</td>
<td>9:15-10:00</td>
<td>NO LECTURE</td>
<td>BN</td>
<td>Hand in March ??</td>
<td>Modelling Exercise:</td>
</tr>
<tr>
<td>5.</td>
<td>Feb 19</td>
<td>0</td>
<td>0</td>
<td>Feb 19</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>BN</td>
<td>testlab</td>
<td>Introduction to testing</td>
</tr>
<tr>
<td>6.</td>
<td>Feb 21</td>
<td>0</td>
<td>0</td>
<td>Feb 21</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>ASk</td>
<td>whitebox blackbox</td>
<td>Classical Test 1+2: (Test case design tekripper I. Whitebox Test case design tekripper II: Blackbox + Coverage)</td>
</tr>
<tr>
<td>7.</td>
<td>Feb 26</td>
<td>0</td>
<td>0</td>
<td>Feb 26</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>BN</td>
<td></td>
<td>Test Driven Development + xUNIT</td>
</tr>
<tr>
<td>8.</td>
<td>Feb 28</td>
<td>0</td>
<td>0</td>
<td>Feb 28</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>BN</td>
<td>fsm-based</td>
<td>Model-Based Testing: (FSM based and UQ test)</td>
</tr>
<tr>
<td>9.</td>
<td>March 6</td>
<td>0</td>
<td>0</td>
<td>March 6</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>AD</td>
<td></td>
<td>Timed Games and Uppaal-TISA</td>
</tr>
<tr>
<td>10.</td>
<td>March 13</td>
<td>0</td>
<td>0</td>
<td>March 13</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>BN</td>
<td></td>
<td>BN Travelling</td>
</tr>
<tr>
<td>11.</td>
<td>March 18</td>
<td>0</td>
<td>0</td>
<td>18 March</td>
<td>9:15-10:00</td>
<td>Group Rooms +PC-Lab</td>
<td>BN</td>
<td>RT-Test</td>
<td>Model-Based Testing: (Online Realtime Uppaal TRON)</td>
</tr>
</tbody>
</table>
# Preliminary Plan

<table>
<thead>
<tr>
<th>No.</th>
<th>Date</th>
<th>SW8</th>
<th>SPZ</th>
<th>Lecture date</th>
<th>Lecture room</th>
<th>Exercise room</th>
<th>Lecturer</th>
<th>Slides</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 March</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Påske</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NO LECTURE</td>
<td></td>
<td></td>
<td>Testing Exercise</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2.12</td>
<td>8.15-10.00</td>
<td>Group Rooms +PC-Lab</td>
<td>EN</td>
<td>Hand in</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>0.2.12</td>
<td>8.15-10.00</td>
<td>Group Rooms +PC-Lab</td>
<td>Andrezej/Urík ?</td>
<td></td>
<td>VisualState I</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td>0.2.12</td>
<td>8.15-10.00</td>
<td>Group Rooms +PC-Lab</td>
<td>Andrezej/Urík ?</td>
<td></td>
<td>VisualState II</td>
</tr>
<tr>
<td></td>
<td>May 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>St. Kr Himmelfart</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td>0.2.12</td>
<td>8.15-10.00</td>
<td>Group Rooms +PC-Lab</td>
<td>Juhan Ernits?</td>
<td></td>
<td>Model Based Testing at Microsoft (with C# and NModel)</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>0.2.12</td>
<td>8.15-10.00</td>
<td>Group Rooms +PC-Lab</td>
<td>Juhan Ernits?</td>
<td></td>
<td>Model Based Testing at Microsoft (with C# and NModel)</td>
</tr>
<tr>
<td>??</td>
<td></td>
<td></td>
<td></td>
<td>8.15-10.00</td>
<td></td>
<td>Group Rooms +PC-Lab</td>
<td>kGL</td>
<td></td>
<td>Probabilistic Modeling &amp; Logics</td>
</tr>
<tr>
<td>??</td>
<td></td>
<td></td>
<td></td>
<td>??</td>
<td>8.15-10.00</td>
<td>Group Rooms +PC-Lab</td>
<td>Guest</td>
<td></td>
<td>Performance Modelling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Probabilistic Model Checking</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW Test in Practice (TK: Validate)</td>
</tr>
</tbody>
</table>

CISS
Plan

- Background
  - Research Group and Projects
- Why (and what) test and verification
- Model-based approach
  - Finite State Machines (review)
  - Interacting State Machines
- Verification=Model Checking (1st glance)
Who are we?
Lecturers

Alexandre David

Brian Nielsen

Arne Skou

... and guests
Research Profile

Distributed Systems & Semantics Unit

Concurrency Theory
Foundation for system behavior

Verification and Validation
Tools for model checking

Networks and Operating Systems
Implementation and construction of platforms

Embedded Systems Methodology
Methods for specification, design, analysis, testing ...
Industrial applications
Why CISS?

- 80% of all software is embedded
- Demands for increased functionality with minimal resources
- Requires multitude of skills
  - Software construction
  - Hardware platforms
  - Control theory
  - Comm. technology
- Goal: Give a qualitative lift to current industrial practice !!!!!!
Partners

- Aeromark
- Analog Devices
- Blip Systems
- Danfoss
- Ericsson Telebit
- ETI
- Exhausto
- FOSS
- GateHouse
- Grundfos
- IAR Systems
- MAN B&W
- Novo Nordisk
- Motorola
- Panasonic
- RTX Telecom
- S-Card
- Simrad
- Skov
- SpaceCom
- TK Systemtest
- TDC Totalløsninger
- Aalborg Industries
Local → Regional → National

DæNES

- Danish Network for Intelligent Embedded Systems

**PARTNERS**

CISS, IMM, MCI, PAJ Systemteknik
GateHouse A/S
ICE Power
Skov A/S
Terma A/S
Novo Nordisk A/S
IO Technologies

- **Funded** by Højteknologifonden

- **Budget** 63 MDKK / 4 years
Local → Regional → National

DaNES

- Selvdiagnostisering og reparation
- Omgivelse
- Indlejret og distribueret kontrol
- Eksekveringsplatform
- Applikation
- Trinvis forfining
- SW API / OS
- HW
- Netværk
- Test og verifikation
- Model domæne
- Fysisk verden
- Udviklingsproces
Quantitative System Properties in Model-Driven-Design of Embedded Systems

Service requirements
- QoS
- Availability
- Fault tolerance

Environment assumptions
- Timing constraints
- Hybrid behavior
- Arrival rates

Communication bandwidth
Computation resources
Power consumption
Costs
Memory usage
A very complex system
Spectacular software bugs
Ariane 5

- The first Ariane 5 rocket was launched in June, 1996. It used software developed for the successful Ariane 4. The rocket carried two computers, providing a backup in case one computer failed during launch. Forty seconds into its maiden flight, the rocket veered off course and exploded. The rocket, along with $500 million worth of satellites, was destroyed.

- Ariane 5 was a much more powerful rocket and generated forces that were larger than the computer could handle. Shortly after launch, it received an input value that was too large. The main and backup computers shut down, causing the rocket to veer off course.
Rotterdam Storm Surge Barrier
Spectacular software bugs
U.S.S. Yorktown, U.S. Navy

- In 1998, the USS Yorktown became the first ship to test the US Navy's Smart Ship program. The Navy planned to use off-the-shelf computers and software instead of expensive custom-made machines. A sailor mistakenly entered a zero for a data value on a computer. Within minutes, Yorktown was dead in the water. It was several hours before the ship could move again.

- When the sailor entered the mistaken number, the computer tried to divide by zero, which isn't possible. The software didn't check to see if the inputs were valid before computing and generated an invalid answer that was used by another computer. The error cascaded several computers and eventually shut down the ship's engines.
The United States established the Ballistic Missile Early Warning System (BMEWS) during the Cold War to detect a Soviet missile attack. On October 5, 1960 the BMEWS radar at Thule, Greenland detected something. Its computer control system decided the signal was made by hundreds of missiles.

The radar had actually detected the Moon rising over the horizon. Unfortunately, the BMEWS computer had not been programmed to understand what the moon looked like as it rose in the eastern sky, so it interpreted the huge signal as Soviet missiles. Luckily for all of us, the mistake was realized in time.
The Therac-25 radiation therapy machine was a medical device that used beams of electrons or photons to kill cancer cells. Between 1985-1987, at least six people got very sick after Therac-25 treatments. Four of them died. The manufacturer was confident that their software made it impossible for the machine to harm patients.

The Therac-25 was withdrawn from use after it was determined that it could deliver fatal overdoses under certain conditions. The software would shut down the machine before delivering an overdose, but the error messages it displayed were so unhelpful that operators couldn't tell what the error was, or how serious it was. In some cases, operators ignored the message completely.
Spectacular Software Bugs
.... continued

- INTEL Pentium II floating-point division
  470 Mill US $

- Baggage handling system, Denver
  1.1 Mill US $/day for 9 months

- Mars Pathfinder

- .......

CISS
CENTER FOR INNOVATIVE SOFTWARE SYSTEMS
Errors in (Embedded) software are extremely expensive

Michael Williams
Research Director, Ericsson, SE
*** STOP: 0x00000000 (0x802aa502,0x00000002,0x00000000,0xF84001C) *** IRQL_NOT_LESS_OR_EQUAL *** Address fa84001c has base at fa840000 - i8042prt.SYS

CPUID: GenuineIntel 5.2.c irql:1f  SYSVER 0xBF0000565

<table>
<thead>
<tr>
<th>Dll Base</th>
<th>Date Stamp - Name</th>
<th>Dll Base</th>
<th>Date Stamp - Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>80100000</td>
<td>2be154c9 - ntoskrnl.exe</td>
<td>80400000</td>
<td>2bc153b0 - hal.dll</td>
</tr>
<tr>
<td>80200000</td>
<td>2bd49628 - ncrc710.sys</td>
<td>80250000</td>
<td>2bd49688 - SCSPORT.SYS</td>
</tr>
<tr>
<td>80267000</td>
<td>2bd49683 - scsidisk.sys</td>
<td>802a6000</td>
<td>2bd496b9 - Fastfat.sys</td>
</tr>
<tr>
<td>fa800000</td>
<td>2bd49666 - Floppy.SYS</td>
<td>fa810000</td>
<td>2bd496db - Npfs_Rec.SYS</td>
</tr>
<tr>
<td>fa820000</td>
<td>2bd49676 - Null.SYS</td>
<td>fa830000</td>
<td>2bd4965a - Beep.SYS</td>
</tr>
<tr>
<td>fa840000</td>
<td>2bdab00 - i8042prt.SYS</td>
<td>fa850000</td>
<td>2bd5a020 - SERMOUSE.SYS</td>
</tr>
<tr>
<td>fa860000</td>
<td>2bd4966f - kbdclass.SYS</td>
<td>fa870000</td>
<td>2bd49671 - MOUSECL.SYS</td>
</tr>
<tr>
<td>fa880000</td>
<td>2bd9c0be - V葶opt.SYS</td>
<td>fa890000</td>
<td>2bd49638 - NCR77C22.SYS</td>
</tr>
<tr>
<td>fa8a0000</td>
<td>2bd494ce - Vga.SYS</td>
<td>fa8b0000</td>
<td>2bd496d0 - Msfs.SYS</td>
</tr>
<tr>
<td>fa8c0000</td>
<td>2bd496c3 - Npfs.SYS</td>
<td>fa8e0000</td>
<td>2bd496c9 - Ntfs.SYS</td>
</tr>
<tr>
<td>fa940000</td>
<td>2bd496df - NDIS.SYS</td>
<td>fa930000</td>
<td>2bd49707 - wdlan.sys</td>
</tr>
<tr>
<td>fa970000</td>
<td>2bd49712 - TDI.SYS</td>
<td>fa950000</td>
<td>2bd5a7fb - nbf.sys</td>
</tr>
<tr>
<td>fa980000</td>
<td>2bd7246 - streams.sys</td>
<td>fa9b0000</td>
<td>2bd4975f - uchnb.sys</td>
</tr>
<tr>
<td>fa9c0000</td>
<td>2bd59fd7 - mcsxms.sys</td>
<td>fa9d0000</td>
<td>2bd4971d - netbios.sys</td>
</tr>
<tr>
<td>fa9e0000</td>
<td>2bd49678 - Parallel.sys</td>
<td>fa9f0000</td>
<td>2bd4969f - serial.SYS</td>
</tr>
<tr>
<td>faa0000</td>
<td>2bd49739 - mmp.sys</td>
<td>faa40000</td>
<td>2bd4971f - SMBTRSUP.SYS</td>
</tr>
<tr>
<td>faa1000</td>
<td>2bd6f2a2 - srv.sys</td>
<td>faa50000</td>
<td>2bd4971a - afd.sys</td>
</tr>
<tr>
<td>faa6000</td>
<td>2bd6f80 - rdr.sys</td>
<td>faaa0000</td>
<td>2bd49735 - browser.sys</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Dword dump Build [1381] - Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe9cdaec</td>
<td>fa84003c fa84003c 00000000 00000000 80149905 - i8042prt.SYS</td>
</tr>
<tr>
<td>fe9cdaf8</td>
<td>8025dfe0 8025dfe0 ff8e6b8c 80129c2c ff8e6b94 - SCSPORT.SYS</td>
</tr>
<tr>
<td>fe9cdbc10</td>
<td>8013e53a 8013e53a ff8e6b94 00000000 ff8e6b94 - ntoskrn1.exe</td>
</tr>
<tr>
<td>fe9cdbc18</td>
<td>8010a373 8010a373 ff8e6df4 ff8e6f60 ff8e6c58 - ntoskrn1.exe</td>
</tr>
<tr>
<td>fe9cdb338</td>
<td>80105683 80105683 ff8e6f60 ff8e6c3c 8015ac7e - ntoskrn1.exe</td>
</tr>
<tr>
<td>fe9cdbc44</td>
<td>80104722 80104722 ff8e6df4 ff8e6f60 ff8e6c58 - ntoskrn1.exe</td>
</tr>
<tr>
<td>fe9cdbc4c</td>
<td>8012034c 8012034c 00000000 80088000 80106fc0 - ntoskrn1.exe</td>
</tr>
</tbody>
</table>
Why T&V?

- Errors in (Embedded) software are extremely expensive
- 30-40% of development time spent on (often ad-hoc) testing.
- There is enormous potential for improved methods and tools.
- “Time-to-market” can be reduced through early verification and performance analysis
Testing vs. Verification
Verification and Test

- Verifikation Kode/Model mht Spec
- Test System mht Model/Spec

System Verification and Test

/* Wait for events */
void OS_Wait(void);
/* Operating system visualSTATE process. Mimics a OS process for a *
visualSTATE system. In this implementation this is the *
interfacing to the visualSTATE basic API. */
void OS_VS_Process(void);
/* Define completion code variable. */
unsigned char cc;
void HandleError(unsigned char ccArg)
{
printf("Error code %c detected, exiting application.\n", ccArg);
exit(ccArg);
}
/* In d-241 we only use the OS_Wait call. It is used to simulate a *
system. It purpose is to generate events. How this is done is up to *
you. */
void OS_Wait(void)
{
/* Ignore the parameters; just retrieve events from the keyboard and *
put them into the queue. When EVENT_UNDEFINED is read from the *
keyboard, return to the calling process. */
SEM_EVENT_TYPE event;
int num;
Test versus Verification

Airbus Control Panel

Beolink

Deadlock identified by
VERIFICATION
after sequence of
2000
msgs / < 1min.

TEST

VERIFICATION

2^n sequences of length n
More complex systems
A simple program

```plaintext
int x=100;

Process INC
  do
    :: x<200 --> x:=x+1
  od

Process DEC
  do
    :: x>0 --> x:=x-1
  od

Process RESET
  do
    :: x=200 --> x:=0
  od

( INC || DEC || RESET )

Which values may x take?

Questions/Properties:
- \( E<> (x>100) \)
- \( E<> (x>200) \)
- \( A[] (x<=200) \)
- \( E<> (x<0) \)
- \( A[] (x>=0) \)
```

Possibly

Always
Another simple program

What are the possible final values of x?

int x=0;
Process P
do
  x:=x+1
10 times
(P || P)

int x=0;
Process P
int r
do
  r:=x; r++; x:=r
10 times
(P || P)

Atomic stm.
Model-based Approach
Suggested Solution?

Model based validation, verification and testing of software and hardware
Traditional Software Development

The Waterfall Model

Problem Area

Analyse

Design

Coding

Testing

♦ Costly in time-to-market and money
♦ Errors are detected late or never
♦ Application of models as early as possible
Introducing, Detecting and Repairing Errors

*Liggesmeyer 98*
Introducing, Detecting and Repairing Errors

Liggesmeyer 98
Model-Driven Development

- Design Model
- Specification
- Verification & Refusal
- Analysis
- Validation
- Implementation
- Testing
- Monitoring

Formal Methods
- UML
- Automatic Code generation
- Automatic Test generation
- Automatic Monitoring

CISS
Real-time Systems

Eg.:
- Realtime Protocols
- Pump Control
- Air Bags
- Robots
- Cruise Control
- ABS
- CD Players
- Production Lines

Real Time System
A system where correctness not only depends on the logical order of events but also on their **timing**!!
Real-time Modeling

Plant
Continuous

Controller Program
Discrete

sensors

actuators

Model of Environment (non-deterministic/User-supplied)

Model of Tasks (user supplied/automatic?)

UPPAAL Model

inputs

outputs
Real-time Model-checking

Plant
Continuous

Controller Program
Discrete

Model of Environment (non-deterministic/User-supplied)

Model of Tasks (user supplied/automatic?)

inputs

outputs

UPPAAL Model

Model of
inputs

outputs

SAT $\phi$ ??

CISS
Real-time Controller Synthesis

Plant
Continuous

Controller Program
Discrete

Model of Environment (non-deterministic/User-supplied)

Partial UPPAAL Model

inputs

outputs

SAT $\phi$ !!

Model of Environment (non-deterministic/User-supplied)

Sensors

Actuators

Synthesis of Tasks/Scheduler (automatic)
Real-time Model-Based Testing

Plant

Continuous

Controller Program

Discrete

sensors

actuators

UPPAAL Model

Test generation (offline or online) wrt. Design Model

inputs

outputs

Conforms-to?

CISS
Real-time Monitoring

Plant
Continuous

Controller Program
Discrete

Observed trace $\sigma \in M$?

Model of Environment (non-deterministic/User-supplied)

Model of Tasks (user supplied/automatic?)

inputs

outputs

UPPAAL Model

CISS
Models

- A model is a simplified representation of the real world.
- Used to gain confidence in the adequacy and validity of a proposed system.
- Models select aspects.
- Removes irrelevant details.

Model

Realization

“implements??”
Models

- Abstractions of the problem-space, not solution space
- Domain Specific Modeling Languages
  - Simulink/StateFlow
  - UML,
- Early exploration of design-alternatives
- Automatic transformation
  - Correctness-by-\textit{construction} vs. Correctness-by-\textit{correction}
Model-based vs. MDD

- **Model Driven Development:**
  - Model is the center of focus from analysis to execution
  - Model is gradually refined / transformed into solution

- **Model-based Development:**
  - (Unrelated) models used to support selected development activities where appropriate
How?

Unified Model = State Machine!
Tamagotchi

ALIVE

Passive → A

Care → A

Medicine → A

Discipline → A

Play → A

Light → A

Feeding:

Meal → A

Snack → B

Health := Health-1

Tick

Tick

Health := Health-1; Age := Age+1

Health=0 or Age=2.000

DEAD

Health:= Health-1
SYNCmaster
Digital Watch
Informationsteknologi

CISS

SPIN, Gerald Holzmann AT&T
**Hierarchical state systems**

**Flat state systems**

**Multiple and inter-related state machines**

**Supports UML notation**

**Device driver access**
Rhapsody
Esterel
NModel

FSM(0,
   AcceptingStates(), Transitions(
      t(0, ShowTitles(), 1),
      t(1, SortByFirst(), 2),
      t(2, SortByMostRecent(), 3),
      t(3, ShowText(), 4)),
   Vocabulary("ShowTitles", "ShowText",
              "SelectMessages", "SelectTopics",
              "SortByFirst", "SortByMostRecent")
)
‘State Explosion’ problem

All combinations = exponential in no. of components

Provably theoretical intractable
Train Simulator

1421 machines
11102 transitions
2981 inputs
2667 outputs
3204 local states
Declare state sp.: $10^{476}$

Our techniques has reduced verification time with several orders of magnitude (ex 14 days to 6 sec)
Modelling and Analysis

Software Model \( \text{A} \)

Requirement \( \text{F} \)

Tools: UPPAAL, visualSTATE, ESTEREL, SPIN, Statemate, FormalCheck, VeriSoft, Java Pathfinder, …
Modelling and Analysis

Software Model

Semantics

 Requirement

Logic

Algorithmics

TOOL

Tools: UPPAAL, visualSTATE,
ESTEREL, SPIN, Statemate, FormalCheck,
VeriSoft, Java Pathfinder,…
Finite State Machines

- Language versus behaviour
- Determinism versus non-determinism
- Composition and operations
- Variants of state machines
  Moore, Mealy, IO automater, UML ....

Most fundamentae model in Computer Science: Kleene og Moore
State Machines

**Model of Computation**
- Set of states
- A **start** state
- An input-alfabet
- A **transition funktion**, mapping input symbols and state to next state
- One ore more **accept** states.
- Computation starts from start state with a given input string (read from left to right)

**Modulo 3 counter**

0, 1, 2, 0, 1, 2, 0, 1,
State Machines

Variants

Machines may have actions/output associated with state—Moore Machines.
State Machines

**Varianter**

Machines may have actions/output associated with med transitions – Mealy Machiner.

Transitions unconditional of input (nul-transitions).

Several transitions for given for input and state (non-determinism).
State Machines

Variants

Symbols of alphabet partitioned in input- and output-actions (IO-automata)
Interacting State Machines
Home-Banking?

int accountA, accountB; //Shared global variables
//Two concurrent bank costumers

Thread costumer1 () {
    int a, b; //local tmp copy
    a = accountA;
    b = accountB;
    a = a - 10; b = b + 10;
    accountA = a;
    accountB = b;
}

Thread costumer2 () {
    int a, b;
    a = accountA;
    b = accountB;
    a = a - 20; b = b + 20;
    accountA = a;
    accountB = b;
}

Are the accounts in balance after the transactions?
Home Banking

A[ ] (pc1.finished and pc2.finished) imply (accountA+accountB==200)?
Home Banking

int accountA, accountB;  //Shared global variables
Semaphore A,B;            //Protected by sem A,B
//Two concurrent bank customers

Thread costumer1 () {
    int a,b;  //local tmp copy
    wait(A);
    wait(B);
    a=accountA;
    b=accountB;
    a=a-10;b=b+10;
    accountA=a;
    accountB=b;
    signal(A);
    signal(B);
}

Thread costumer2 () {
    int a,b;
    wait(B);
    wait(A);
    a=accountA;
    b=accountB;
    a=a-20; b=b+20;
    accountA=a;
    accountB=b;
    signal(B);
    signal(A);
}
Semaphore FSM Model

Binary Semaphore

Counting Semaphore

- open
- wait?
- closed

- signal?

- c:=init_count
- c>0
- c:=c-1
- wait?
- c:=c+1
- counting
- signal?
Composition

IO Automater (2-vejs synkronisering)
Composition

IO Automater (2-vejs synkronisering)
Semaphore Solution?

1. Consistency? (Balance)
2. Race conditions?
3. Deadlock?

1. $A[]$ (mc1.finished and mc2.finished) imply (accountA+accountB==200) ✓
2. $E<> mc1.critical\_section and mc2.critical\_section$ ✓
3. $A[]$ not (mc1.finished and mc2.finished) imply not deadlock ✗