RoboChart and RoboSim
Verified Simulation for Robotics

Jim Woodcock FREng

University of York, UK — RoboStar
www.cs.york.ac.uk/RoboStar

13th June 2019

Thanks:
Ana Cavalcanti, James Baxter, Madiel Conserva, André Didier, Simon Foster, Wei Li,
Alvaro Miyazawa, Alexandre Mota, Pedro Ribeiro, Augusto Sampaio, Jon Timmis,
Current approach to development

1st phase: Abstract model

*state machine*

2nd phase: Simulation

<table>
<thead>
<tr>
<th>controller code</th>
<th>hardware simulation</th>
<th>discrete environment simulation</th>
</tr>
</thead>
</table>

3rd phase: Implementation

<table>
<thead>
<tr>
<th>low-level code</th>
<th>robot</th>
<th>environment</th>
</tr>
</thead>
</table>
Example state machines

<table>
<thead>
<tr>
<th>Chain</th>
<th>Explore</th>
<th>Navigate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td>(Recruitment</td>
</tr>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td>from below)</td>
</tr>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td>(Recruitment</td>
</tr>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td>from above)</td>
</tr>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td>navigate down</td>
</tr>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td>navigate up</td>
</tr>
<tr>
<td>Recruiting?</td>
<td>Recruitable?</td>
<td></td>
</tr>
</tbody>
</table>

Becky Naylor, Mark Read, Jon Timmis, and Andy Tyrrell.
The Relay Chain: Communication between an Exploratory Underwater Shoal and a Surface Vehicle.
Our vision

RoboChart and RoboSim Verified Simulation for Robotics

- RoboEnv+RoboChart Model
  - AUTOMATIC GENERATION
  - Test cases
  - CONVERSION

- Proof Model
  - AUTOMATIC GENERATION

- Properties of interest
  - Model checkers
    - Theorem provers
  - valid?

- RoboSim Model
  - AUTOMATIC GENERATION

- Robotics Simulator
  - AUTOMATIC GENERATION
  - valid?

- Deployment Code
  - AUTOMATIC GENERATION
  - YES
  - NO

- System Testing
  - CERTIFIED FOR SAFE USE
  - Code Proofs
  - Test results and assumptions
  - Environment restrictions
  - correct?
  - YES
  - NO

- Test cases
  - YES
  - NO
RoboChart

- Domain-specific modelling language for roboticists
- Based on UML, but restricted…
  - State machines
  - Component model
  - Physical robot explicitly modelled
- … and enriched
  - time properties
  - probabilistic choices
- Simplified compositional semantics and automated reasoning
  - Model checking
  - Theorem proving
- Process algebraic semantics
- RoboTool: modelling, well formedness, and more
RoboTool

The diagram illustrates the integration of RoboChart, RoboSim, and RoboTool for verified simulation in robotics. It shows how requirements are used to create RoboChart models, which are then formalized using Reactive Modules Formalism and CSP and timed-CSP. The formal models are then translated to C++ and used with tools like PRISM and Storm for qualitative results. Quantitative results are obtained through simulation using ARGoS.
RoboChart Models

Module

- Identifies a robotic system
- Single robotic platform
- Interaction with robotic platform is via
  - Shared variables
  - Operations
  - Events
- One or more controllers
- Communication is asynchronous
Controller

- Models a specific behaviour
- Contains:
  - Variables
  - Operations
  - Events
  - Behaviour: *state machines*
- Multiple state-machines can be used to define behaviour.
- Communication between controllers
  - Synchronous
  - Asynchronous
- Communication between state machines is synchronous.
State machines

- Self-contained
  - Variables
  - Operations
  - Events
- Well defined action language
- No interlevel transitions
- Urgent transitions
- Time primitives: timed automata + Timed CSP
  - Clock
  - Delay: wait
  - Deadlines over events and actions
- Probabilistic choices
RoboSim

- There are several (commercial) simulators.
  - Netlogo, MASON, JASON, ...
  - Argos, Enki, ...
  - Webots, Microsoft Robotics Developer Studio
  - Simulink + Stateflow
- In all cases, cyclic simulations.
- RoboChart is not a simulation language.
- Based on interrupts: events handled when available.
- No notion of cycle
- We still want a diagrammatic notation for simulation.
- We want to relate RoboChart models to RoboSim models.
**Buffered parallelism**

Visible behaviour: registerRead and registerWrite
What is a correct simulation?

Criterion

- Its behaviour is a possible behaviour of the RoboChart model.
- Behaviour
  - Sequence of interactions
  - Refusal to interact
  - Livelock
  - Time
- Interaction
  - Read and write to global variables
  - Operation calls
  - Events
What is a correct simulation?

Summary

- RoboChart describes order, availability, and time of accesses to global vars, events, operation calls of platform.
- RoboSim describes order, availability, and time of registerRead and registerWrite register in a cycle.

We need to relate

- registerRead.true to obstacle
- registerWrite.true to a call to move
- Interrupt view to a cyclic view
What is a correct simulation?

We compare RoboChart + Assumptions...

- Calculated model of RoboChart
- Assumptions:
  - Connection of input events to \textit{registerRead} events
  - Connection of operation calls and output events to \textit{registerWrite} events
  - Events do not occur between cycles.
  - Each cycle has at most one occurrence of each output.
  - In each cycle, each operation is called at most once.

...with RoboSim model.

- Calculated model of RoboSim
Mistakes we can find

We cannot write a simulation for this machine: missing wait.
Schedulability

- Our technique can determine schedulability of a given cycle.
  - If schedulable:
    - Intuitively, the cycle needs to be at most the greatest common divisor of all time restrictions.
    - It needs to be a common divisor.
    - The assumptions are feasible.
  - We can verify with a deadlock check
    
    *RoboChart + Assumptions is deadlock free.*
Mistakes we can find

Missing entry action
Mistakes we can find

Two moves in a single step

![Diagram showing a RoboChart and RoboSim verified simulation for robotics with a focus on mistakes such as two moves in a single step.]
Mistakes we can find

Missing abstract (RoboChart) event
Mistakes we can find

Missing treatment of obstacle

Simulation_SimFW [cycle = 1]

Inputs          Outputs

I Sensors  I Operations

exec [not $obstacle]/#MBC

exec $move(lv, 0)

exec [since(MBC) >= PI/av]

exec [since(MBC) < PI/av]

DMoving

SMoving

exec [obstacle]

DTurning

STurning

entry $move(lv, 0)

entry $move(0, av)

move(ls: real, as: real)

obstacle

I Sensors

obstacle
Mistakes we can find

Possibility of ignoring abstract event

<table>
<thead>
<tr>
<th>Simulation_SimFW [cycle = 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI: real, av: real, lv: real</td>
</tr>
<tr>
<td>MBC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISensors</td>
<td>IOperations</td>
</tr>
</tbody>
</table>

```
exec

SMoving
entry $move(lv, 0)

exec [since(MBC)<PI/av]

DMoving
exec [$obstacle]/#MBC

exec [not $obstacle]

exec [since(MBC) >= PI/av]

DTurning

STurning
entry $move(0, av)
```

IOperations
- move(ls: real, as: real)

ISensors
- obstacle

obstacle
So, what next?

Currently

- Case studies: driverless pod, sandwich maker, …
- Automatic translation from RoboChart to RoboSim
- Physical models for platforms
- Models for the environment
- Test generation
- …
So, what next?

A lot to do

- Computer vision, artificial intelligence, human-robot interaction, ethics, safety cases, ...

- **Software Engineering**
  - Theory: UTP
  - Practice: new languages (formal, diagrammatic, API)
  - Security, real-time analysis

Our distinctive vision

- Notations akin to those already used
- Sound integration
- Full life cycle

The theory is that of cyber-physical systems.