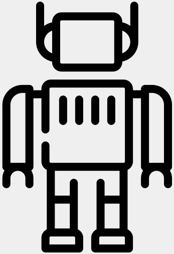
RoboChart & RoboSim

Modelling Robots and Collections

Alvaro Miyazawa

Department of Computer Science
University of York

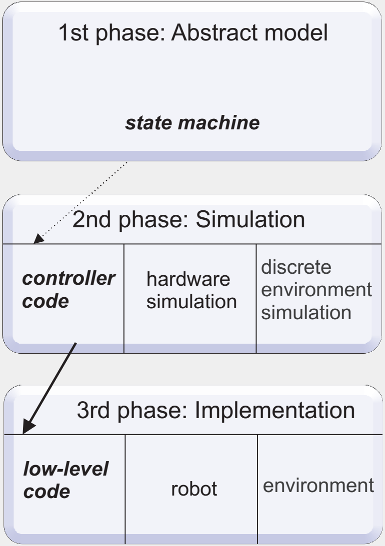
January 23, 2019



OUTLINE

- Introduction
- RoboChart
- RoboSim
- Collection modelling
- Robotic platform modelling

Introduction

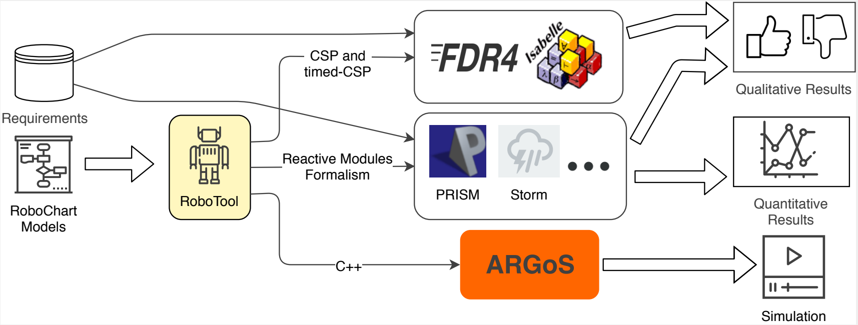

1st phase: Abstract model
state machine
2nd phase: Simulation
controller code
hardware simulation
discrete environment simulation
3rd phase: Implementation
low-level code
robot
environment

Motivation

- State machines are often used to record, illustrate and explain
- Usage is informal
- Potential:
  - Testing
  - Code generation
  - Verification

- Graphical notations
- Formal semantics
- Specialised, but comprehensive
- Supporting simulation, analysis and verification

Approach



Requirements
RoboChart
Models
RoboTool
CSP and
timed-CSP
Reactive Modules
Formalism
C++
FDR4
Isabelle
PRISM
Storm
ARGoS
Qualitative Results
Quantitative
Results
Simulation

RoboChart

- Standard state machines + time + probability
- Formal semantics: untimed, timed and probabilistic
- Well-formedness conditions
- Tool support:
  - Modelling
  - Validation
  - Code generation: semantics and simulation

Module

- Models a single Robot
- 1 Robotic Platform
- 1+ Controllers
- Communication
  - Synchronous
  - Asynchronous
- Robotic Platform may provide shared variables

- Records assumptions about the robot hardware
  - which events the robot provides
  - which operations the robot supports
  - which variables are available
- Independent of controller and state-machines
- Single point of interaction with robot

- Models a specific behaviour
- Contains:
  - Behavioural state-machines
  - Operations
  - Variables
  - Events
- Supports multiple behavioural state-machines
- Communication between state-machines is synchronous

State-Machine

- Main behavioural specification construct
- Models both operations and behaviours
- Simple, Composite and Final states
- Initial and junction nodes
- Non-interlevel transitions
- Actions: entry, during, exit, transition
- Local variables

Types and Action Language

- Types based on Z Mathematical Toolkit
- Action language:
  - Assignment
  - Event signalling
  - Operation call
  - Sequential composition
- Control statements modelled using junctions and transitions

Semantics

- Formalised in CSP
- Coverage:
  - State-Machines
  - Controllers
  - Robotic Platforms
  - Modules

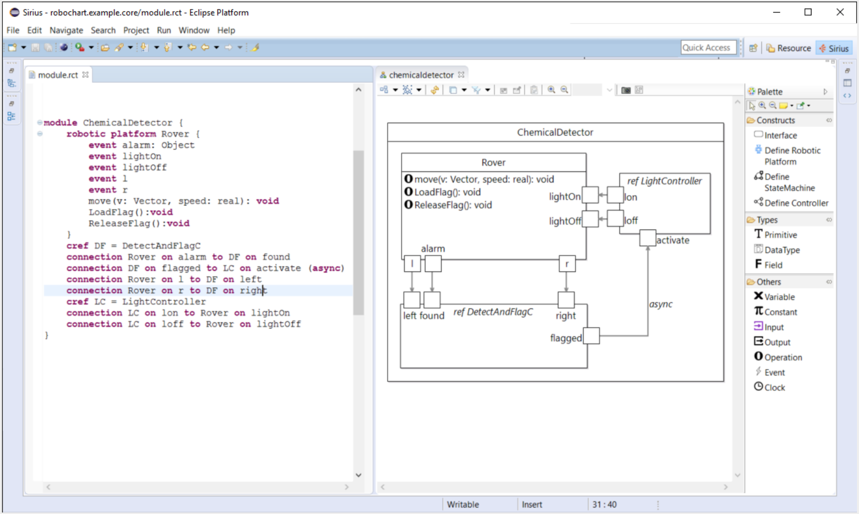
- Module = CSP Process
  - Parallel composition of controllers
  - Connections define synchronisation sets
  - Asynchronous communication modelled through buffers
  - Robotic platform incorporated via renaming
- Controller = CSP Process
  - Parallel composition of state-machines
  - Connections define synchronisation sets
  - External interactions via controller established via renaming

- State-Machine = CSP Process
  - Parallel composition of states
  - Transitions are part of the source states
  - Junctions are part of the incoming transition
  - Initial nodes and final states are part of the parent state
  - States interact with each other to enter and exit
  - States synchronise on transition triggers to support top-down interruption
- Action language
  - Operation call = Process call
  - Event signalling = Communication on event channel
  - Assignment = Communication on setter channel
- State components
  - Isolated in memory process due to sharing
  - Help avoid polling for transition conditions

RoboTool

- Eclipse plugins
- Textual editor developed using Xtext
- Graphical editor developed using Sirius
- Code generator for the semantics
- Code generator for simulation
- Validation rules

RoboTool



Sirius - robochart.example.core/module.rct - Eclipse Platform
File Edit Navigate Search Project Run Window Help
module.rct
module ChemicalDetector {
robotic platform Rover {
event alarm: Object
event lightOn
event lightOff
event l
event r
move(v: Vector, speed: real): void
LoadFlag():void
ReleaseFlag():void
}
cref DF = DetectAndFlagC
connection Rover on alarm to DF on found
connection DF on flagged to LC on activate (async)
connection Rover on l to DF on left
connection Rover on r to DF on right
cref LC = LightController
connection LC on lon to Rover on lightOn
connection LC on loff to Rover on lightOff
}
chemicaldetector
ChemicalDetector
Rover
move(v: Vector, speed: real): void
LoadFlag(): void
ReleaseFlag(): void
lightOn
lightOff
alarm
ref LightController
lon
loff
activate
async
left found
ref DetectAndFlagC
right
flagged
Palette
Constructs
Interface
Define Robotic Platform
Define StateMachine
Define Controller
Types
Primitive
DataType
Field
Others
Variable
Constant
Input
Output
Operation
Event
Clock
Quick Access
Resource
Sirius
Writable
Insert
31 : 40

- Case studies:
  - Alpha Algorithm (Single Robot and Collection);
  - Chemical Detector;
  - Autonomous Chemical Detector;
  - Foraging;
  - Transport; etc.
- Generated semantics used for verification using FDR4
- FDR4 compression functions highly effective

Current developments

- Generation of simulations
- Generation of probabilistic semantics
- Generation of sematics for Isabelle/UTP

RoboSim

- Based on RoboChart
- Explicit cyclic pattern for simulation
- Related to RoboChart models via refinement

Collection Modelling

Motivation

RoboChart

The focus of RoboChart is the modelling, analysis and simulation of individual robots.

RoboChart

The focus of RoboChart is the modelling, analysis and simulation of individual robots.

Other notations

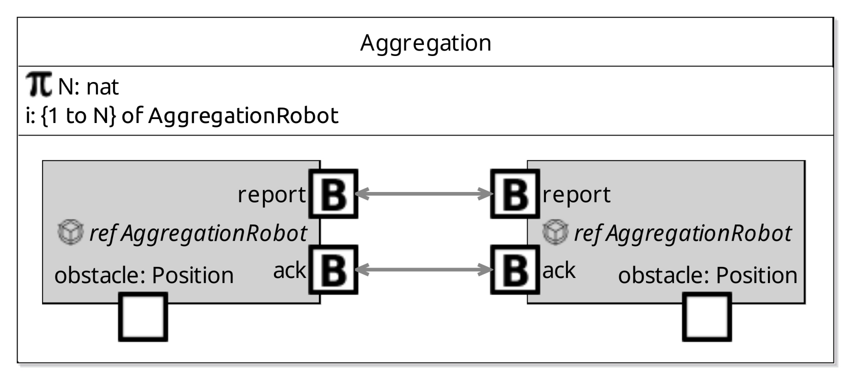
Support in other notations tends to be concrete.

Objective

- Support modelling, analysis and simulation of collections
- Reuse RoboChart models and semantics

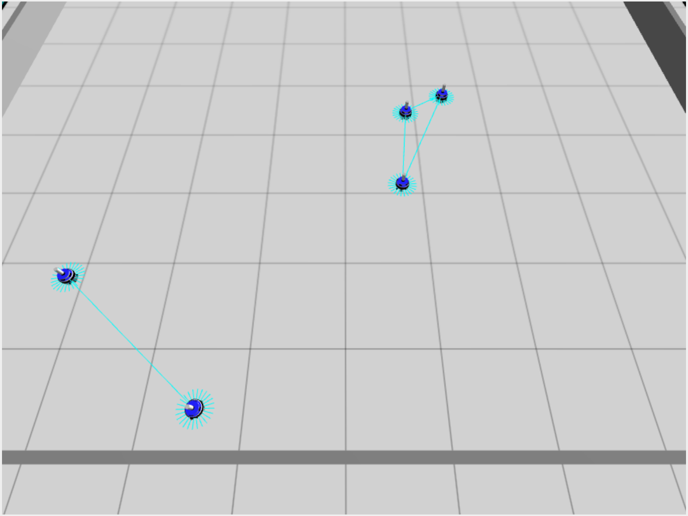
Extensions

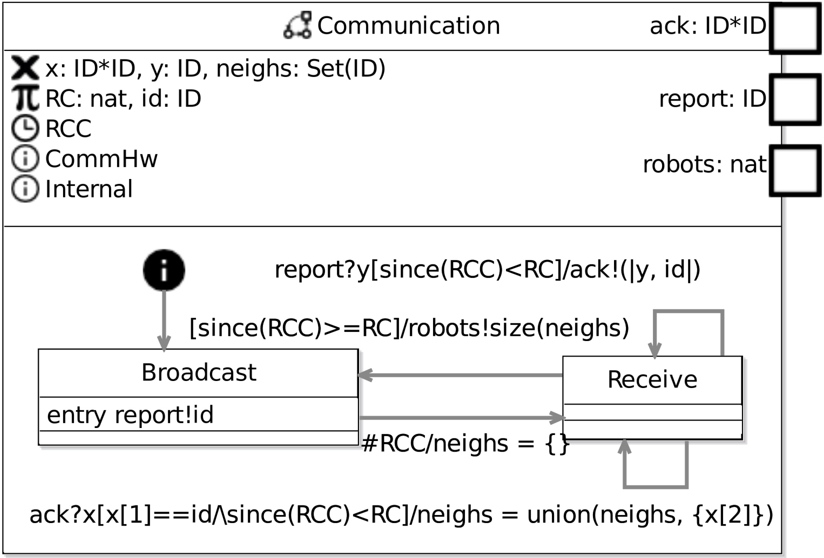
- new implicit type `ID` and module constant `id`;
- robotic platform events are broadcast and directional;
- broadcast events have implicit `ID` parameters: `to` and `from`;
- input events can restrict `from` and record its value;
- output events can restrict `to` parameter; and
- new diagram describes group of collections and how they communicate.


Aggregation
π N: nat
i: {1 to N} of AggregationRobot
report
ref AggregationRobot
obstacle: Position
ack
B
B
B
B
report
ref AggregationRobot
ack
obstacle: Position

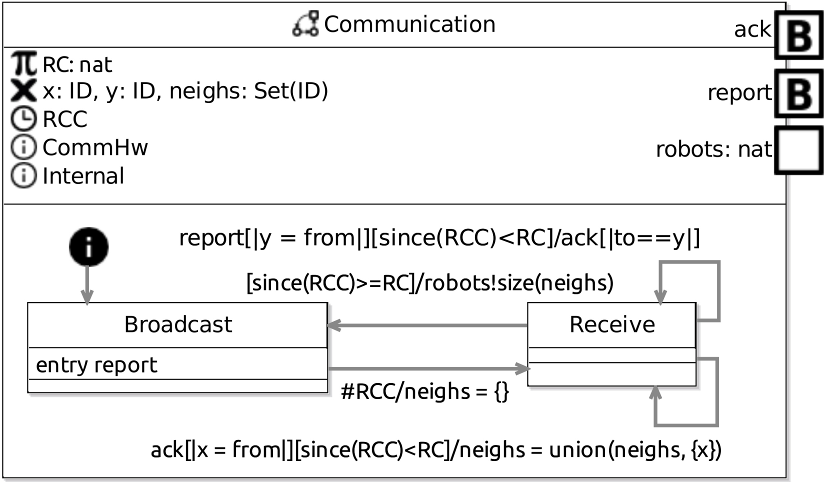
$$
\begin{array}{l}
(|||\, i : \{1..N\} \bullet \mathit{AggregationRobot}(i)) \\
\quad [| \{| \mathit{report.in}, \mathit{report.out}, \mathit{ack.in}, \mathit{ack.out} |\} |] \\
\left(\begin{array}{l}
|||\, i : \{1..N\} \bullet |||\, j : (\{1..N\} \setminus \{i\}) \bullet \mathit{Buffer}(\langle\rangle, \mathit{report}, i, \mathit{report}, j) \\
||| \\
|||\, i : \{1..N\} \bullet |||\, j : (\{1..N\} \setminus \{i\}) \bullet \mathit{Buffer}(\langle\rangle, \mathit{ack}, i, \mathit{ack}, j)
\end{array}\right)
\end{array}
$$

Alpha Algorithm




Communication
ack: ID*ID
x: ID*ID, y: ID, neighs: Set(ID)
RC: nat, id: ID
RCC
CommHw
Internal
report: ID
robots: nat
report?y[since(RCC)<RC]/ack!(|y, id|)
[since(RCC)>=RC]/robots!size(neighs)
Broadcast
entry report!id
Receive
#RCC/neighs = {}
ack?x[x[1]==id/\since(RCC)<RC]/neighs = union(neighs, {x[2]})

Alpha Algorithm (new)



- `ev![|pred|]!e`

  semantics $ev.out.id?to : \{x \mid x \leftarrow ID, pred\}!e \longrightarrow Skip$

- `ev[| v = from | pred |]?u`

  semantics $ev.in?from : \{x \mid x \leftarrow ID, pred\}.id?y \longrightarrow$
  $set\_v!from \longrightarrow set\_u!y \longrightarrow Skip$

Current status

- Partial support for modelling
- Code generation for semantics
- Validation

Current status

- Partial support for modelling
- Code generation for semantics
- Validation

Ongoing work

- Complete modelling support
- Extend simulation generation

Current status

- Partial support for modelling
- Code generation for semantics
- Validation

Ongoing work

- Complete modelling support
- Extend simulation generation

Future work

- Optimise verification
- Investigate data abstraction and induction with FDR4
- Investigate theorem proving with Isabelle/UTP

Robotic platform modelling

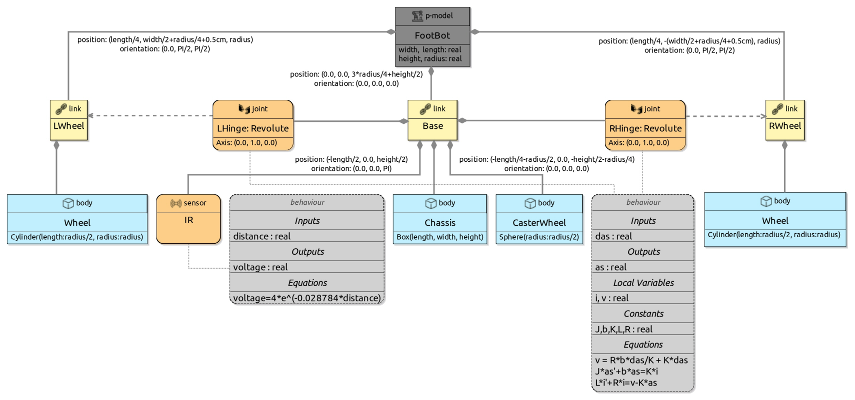
Motivation

- RoboChart focuses on modelling controllers
- Robotic platform is abstracted as a set of events, variables and operations
- Existing XML-based notations: URDF, SDF, Collada
  - not convenient for modelling
  - not abstract enough
  - no facilities for modelling behaviour

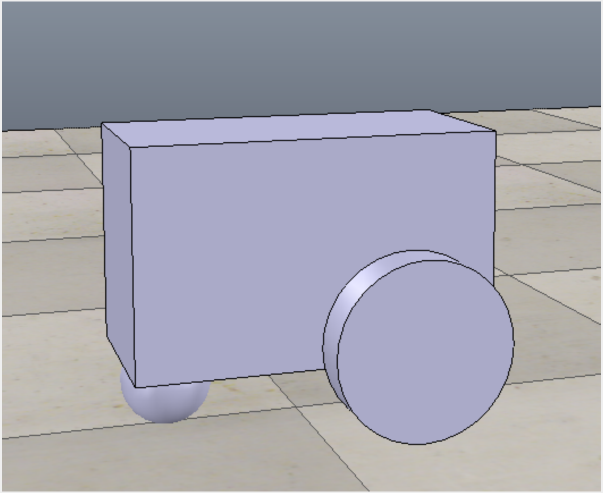
Objectives

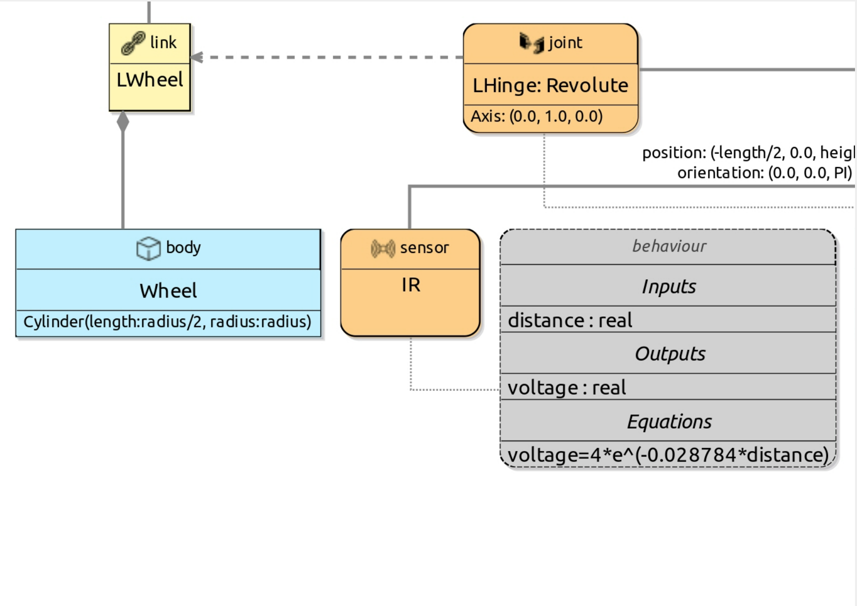
- Restructure and refactor SDF
- Provide graphical representation
- Extend with facilities to
  - model behaviours
  - map between operations, events and variables to sensors and actuators
- Formal semantics integrated with RoboSim
- Linked to RoboChart via abstraction
- Generate both SDF models and platform dependent simulation code

Simple Model

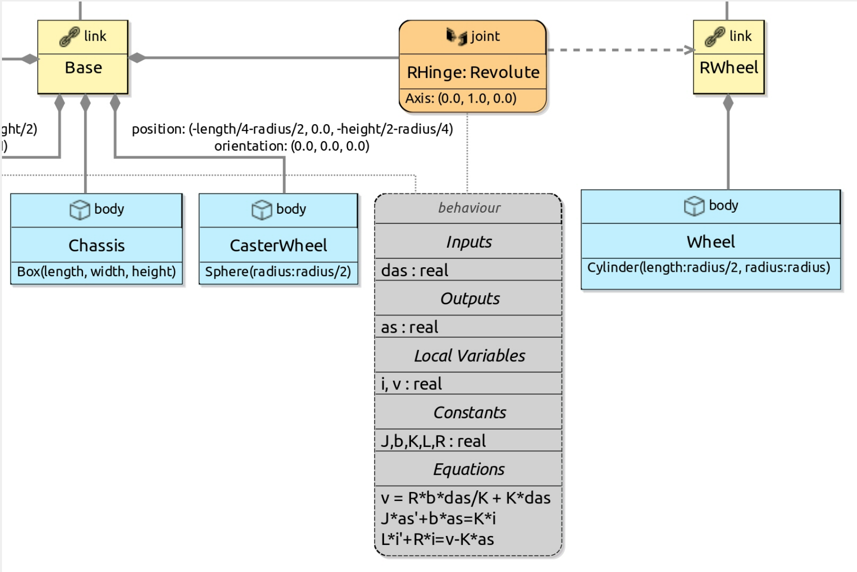


Simple Model

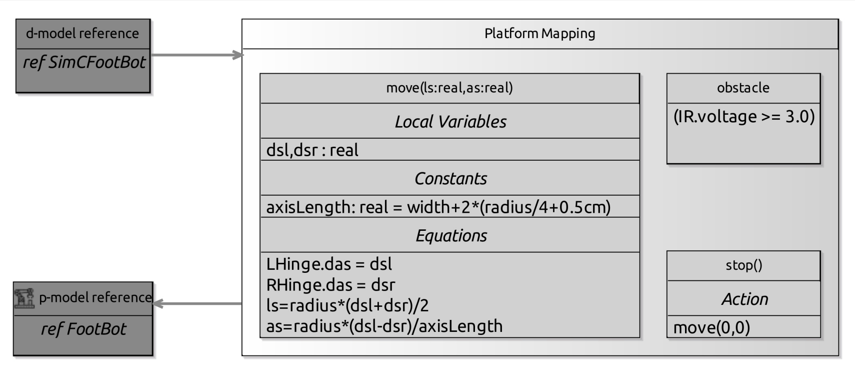



link
LWheel
joint
LHinge: Revolute
Axis: (0.0, 1.0, 0.0)
position: (-length/2, 0.0, heig
orientation: (0.0, 0.0, PI)
body
Wheel
Cylinder(length:radius/2, radius:radius)
sensor
IR
behaviour
Inputs
distance : real
Outputs
voltage : real
Equations
voltage=4*e^(-0.028784*distance)

Simple Model



Simple Model



SEMANTICS

Inputs

$distance : \mathcal{T} \to \mathcal{R}$

Outputs

$las, ras : \mathcal{T} \to \mathcal{R}$

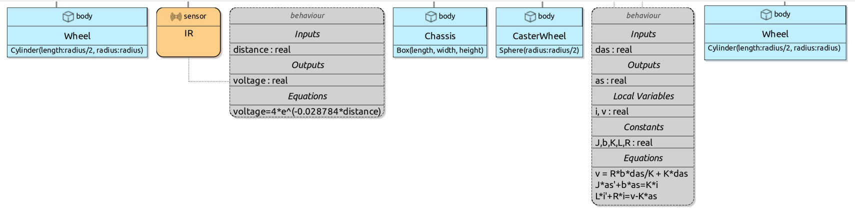
Behaviour *Revolute*

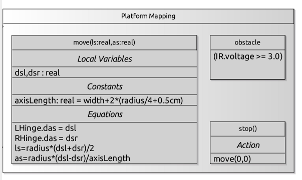
$$v = R \times b \times das/K + K \times das$$
$$J \times as' + b \times as = K \times i$$
$$L \times i' + R \times i = v - K \times as$$

Behaviour *IR*

$$voltage = 4 \times e^{-0.028 \times distance}$$

Simple Model



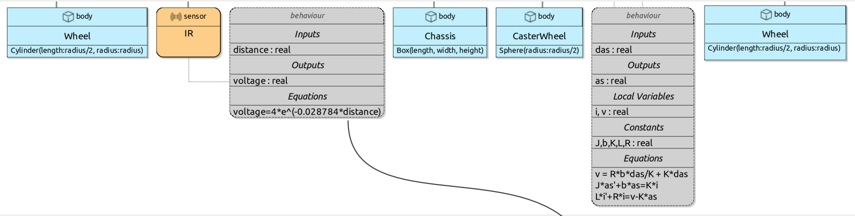


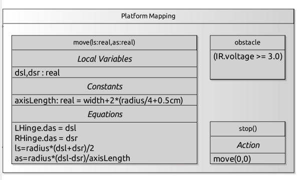
$$\boxed{\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)}$$

$$\mathcal{S}tep(l, r) = \mu X \bullet \begin{pmatrix} (\mathcal{A}\ \mathbf{init}\ ldas, rdas = l, r) \\ \mathbf{until}\ (voltage > 3);\ obstacle \longrightarrow X \end{pmatrix}$$

$$\mathcal{M} = \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet \mathcal{S}tep(l, r) \triangle \left(move.ls.as \longrightarrow \{l, r\} : \begin{bmatrix} true, & ls = rd \times (l + r)/2 \wedge \\ & as = rd \times (l - r)/aL \end{bmatrix};\ X \right)$$

Simple Model





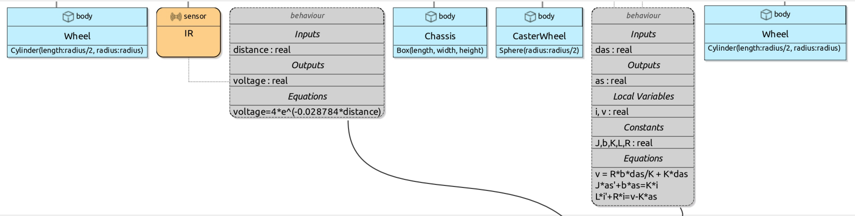
$$\boxed{\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)}$$

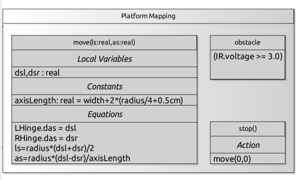
$$\mathcal{S}tep(l, r) = \mu X \bullet \left(\begin{array}{l} (\mathcal{A}\ \mathbf{init}\ ldas, rdas = l, r) \\ \mathbf{until}\ (voltage > 3);\ obstacle \longrightarrow X \end{array} \right)$$

$$\mathcal{M} = \ \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet \mathcal{S}tep(l, r) \triangle$$

$$\left(move.ls.as \longrightarrow \{l, r\} : \left[\begin{array}{ll} true, & ls = rd \times (l + r)/2 \wedge \\ & as = rd \times (l - r)/aL \end{array} \right] ;\ X \right)$$

Simple Model



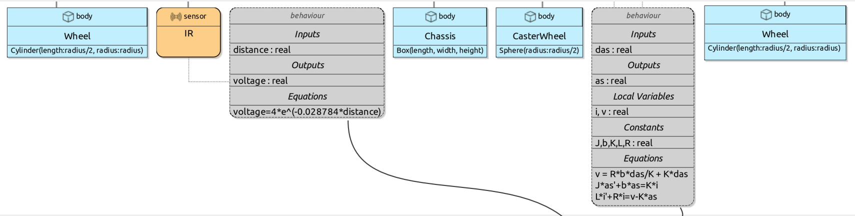


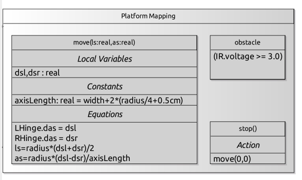
$$\boxed{\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)}$$

$$\mathcal{S}tep(l, r) = \mu X \bullet \begin{pmatrix} (\mathcal{A}\ \mathbf{init}\ ldas, rdas = l, r) \\ \mathbf{until}\ (voltage > 3);\ obstacle \longrightarrow X \end{pmatrix}$$

$$\mathcal{M} = \ \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet \mathcal{S}tep(l, r) \bigtriangleup \left(move.ls.as \longrightarrow \{l, r\} : \begin{bmatrix} true, & ls = rd \times (l + r)/2 \land \\ & as = rd \times (l - r)/aL \end{bmatrix} ; X \right)$$

Simple Model



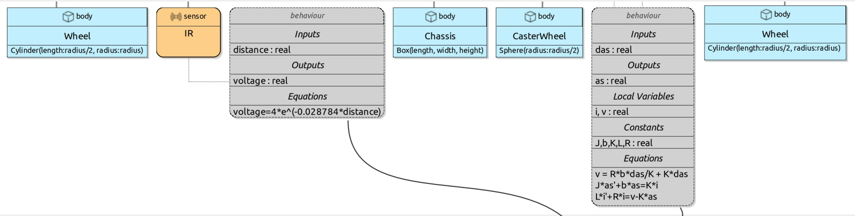


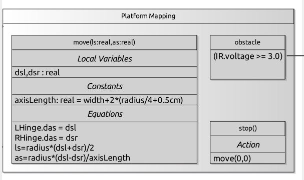
$$\boxed{\mathcal{A} = (\mathit{Revolute}[\![\mathit{das} := \mathit{ldas}, \ldots]\!] \mid \mathit{Revolute}[\![\mathit{das} := \mathit{rdas}, \ldots]\!] \mid \mathit{IR})}$$

$$\mathit{Step}(l, r) = \mu X \bullet \left(\begin{array}{l} (\mathcal{A}\ \mathbf{init}\ \mathit{ldas}, \mathit{rdas} = l, r) \\ \boxed{\mathbf{until}\ (\mathit{voltage} > 3);\ \mathit{obstacle} \longrightarrow X} \end{array} \right)$$

$$\mathcal{M} = \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet \mathit{Step}(l, r) \triangle \left(\mathit{move.ls.as} \longrightarrow \{l, r\} : \left[\begin{array}{ll} \mathit{true}, & \mathit{ls} = \mathit{rd} \times (l + r)/2 \wedge \\ & \mathit{as} = \mathit{rd} \times (l - r)/\mathit{aL} \end{array} \right];\ X \right)$$

Simple Model



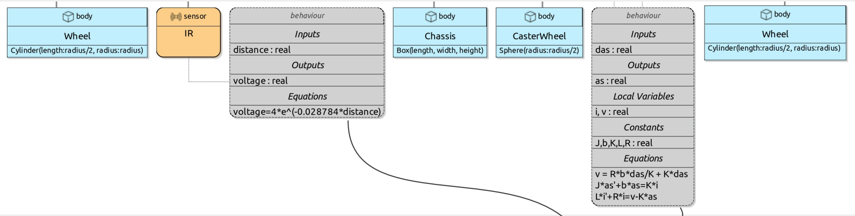


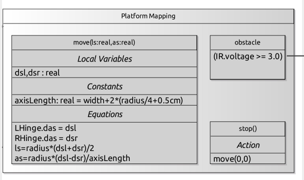
$$\boxed{\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)}$$

$$Step(l,r) = \mu X \bullet \left((\mathcal{A}\ \mathbf{init}\ ldas, rdas = l, r)\ \boxed{\mathbf{until}\ (voltage > 3);\ obstacle \longrightarrow X} \right)$$

$$\mathcal{M} = \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet Step(l,r) \bigtriangleup \left(move.ls.as \longrightarrow \{l, r\} : \left[true,\ \begin{array}{l} ls = rd \times (l+r)/2 \wedge \\ as = rd \times (l-r)/aL \end{array} \right];\ X \right)$$

Simple Model



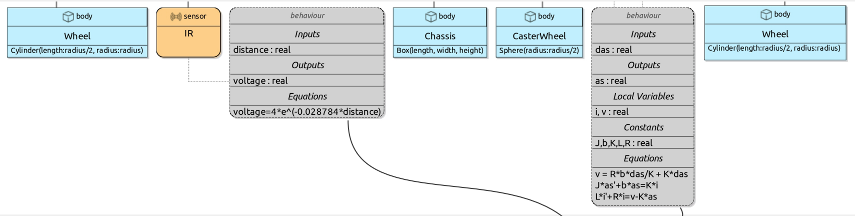


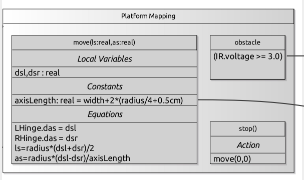
$$\boxed{\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)}$$

$$Step(l,r) = \mu X \bullet \left((\mathcal{A}\ \mathbf{init}\ ldas, rdas = l, r)\ \boxed{\mathbf{until}\ (voltage > 3);\ obstacle \longrightarrow X} \right)$$

$$\mathcal{M} = \ \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet Step(l,r) \triangle \left(\boxed{move.ls.as \longrightarrow \{l, r\} : \left[\begin{array}{ll} true, & ls = rd \times (l+r)/2 \wedge \\ & as = rd \times (l-r)/aL \end{array} \right]} ;\ X \right)$$

Simple Model





$$\boxed{\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)}$$

$$Step(l, r) = \mu X \bullet \left(\begin{array}{l} (\mathcal{A}\ \mathbf{init}\ ldas, rdas = l, r) \\ \boxed{\mathbf{until}\ (voltage > 3);\ obstacle \longrightarrow X} \end{array} \right)$$

$$\mathcal{M} = \mathbf{var}\ l, r : \mathcal{R} \bullet l, r := 0, 0;\ \mu X \bullet Step(l, r) \triangle \left(\boxed{move.ls.as \longrightarrow \{l, r\} : \begin{bmatrix} true, & ls = rd \times (l + r)/2 \wedge \\ & as = rd \times (l - r)/aL \end{bmatrix}} ;\ X \right)$$

Behaviours

$$\mathcal{A} = (\mathit{Revolute}[\![\mathit{das} := \mathit{ldas}, \ldots]\!] \mid \mathit{Revolute}[\![\mathit{das} := \mathit{rdas}, \ldots]\!] \mid \mathit{IR})$$

Behaviours

$$\mathcal{A} = (\mathit{Revolute}[\![\mathit{das} := \mathit{ldas}, \ldots]\!] \mid \mathit{Revolute}[\![\mathit{das} := \mathit{rdas}, \ldots]\!] \mid \mathit{IR})$$

$$\mathcal{S}\mathit{tep}(l, r) = \mu\, X \bullet \begin{pmatrix} (\mathcal{A}\, \textbf{init}\, \mathit{ldas}, \mathit{rdas} = l, r)\, \textbf{until}\, (\mathit{voltage} > 3); \\ \mathit{obstacle} \longrightarrow X \end{pmatrix}$$

Behaviours

$$\mathcal{A} = (Revolute[\![das := ldas, \ldots]\!] \mid Revolute[\![das := rdas, \ldots]\!] \mid IR)$$

$$Step(l, r) = \mu X \bullet \begin{pmatrix} (\mathcal{A}\,\textbf{init}\,ldas, rdas = l, r)\,\textbf{until}\,(voltage > 3); \\ obstacle \longrightarrow X \end{pmatrix}$$

$$\mathcal{M} = \textbf{var}\,l, r : \mathcal{R} \bullet l, r := 0, 0;$$

$$\mu X \bullet \left(\begin{array}{l} Step(l, r) \bigtriangleup move.ls.as \longrightarrow \\ \{l, r\} : \left[true, \begin{array}{l} ls = rd \times (l + r)/2 \wedge \\ as = rd \times (l - r)/aL \end{array} \right]; X \end{array} \right)$$

- $\mathcal{A}$: behaviours of the platform model.
- $\mathcal{S}tep$: behaviours in $\mathcal{A}$ until input events are true.
- $\mathcal{M}$: behaviours in $\mathcal{S}tep$ interrupted by variables assignments, operation calls and output events

Conclusions

- RoboChart supports modelling including time and probability
- Formal semantics specified in CSP
- Tool support for modelling, verification and simulation
- RoboSim models can be
  - derived from RoboChart models
  - related to RoboChart models formally
- Partial support for modelling collections and robotic platforms

Current work

- Modelling support for platform modelling
- Case studies in platform modelling
- Generation of
  - SDF models
  - simulation code
  - formal semantics
- Integration with RoboChart models via abstraction

References

Ana Cavalcanti, Alvaro Miyazawa, Augusto Sampaio, Wei Li, Pedro Ribeiro, and Jon Timmis.
Modelling and verification for swarm robotics.
In Carlo A. Furia and Kirsten Winter, editors, *Integrated Formal Methods*, pages 1–19, Cham, 2018. Springer International Publishing.
DOI: 10.1007/978-3-319-98938-9_1.

Alvaro Miyazawa, Pedro Ribeiro, Wei Li, Ana Cavalcanti, Jon Timmis, and Jim Woodcock.
RoboChart: modelling and verification of the functional behaviour of robotic applications.
Software and Systems Modeling, 2019.
DOI: 10.1007/s10270-018-00710-z (To Appear).