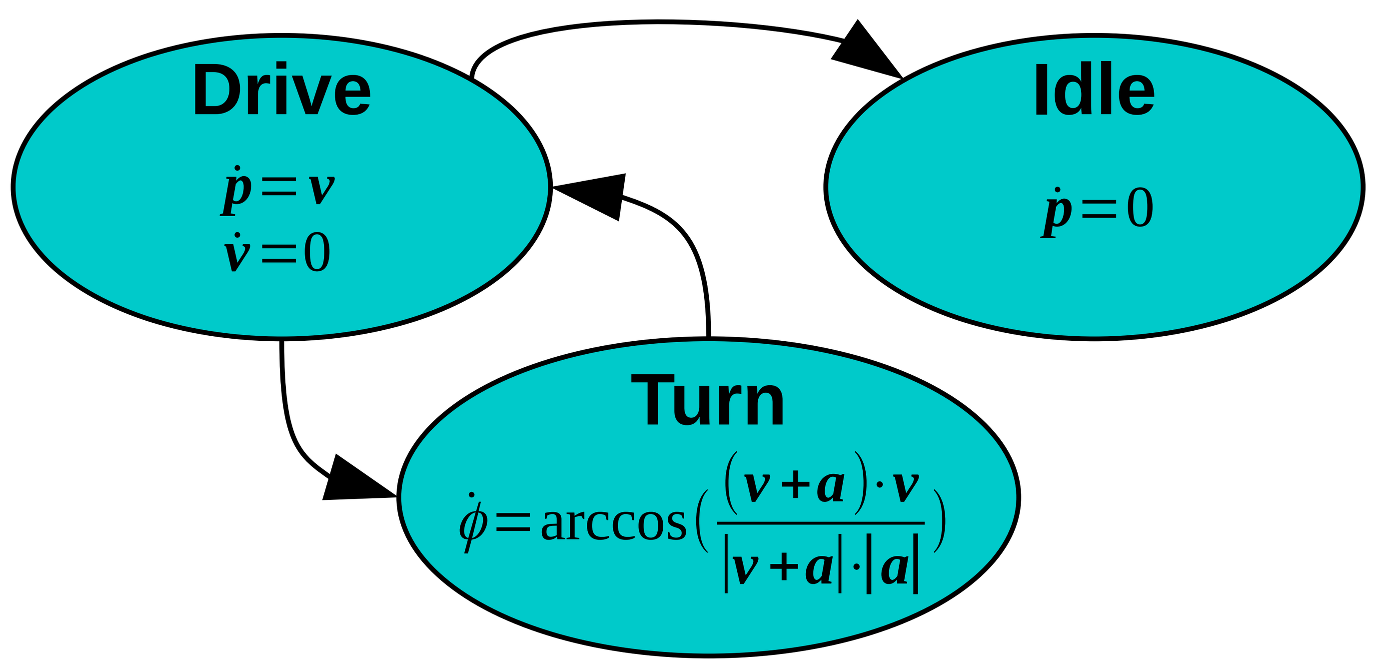
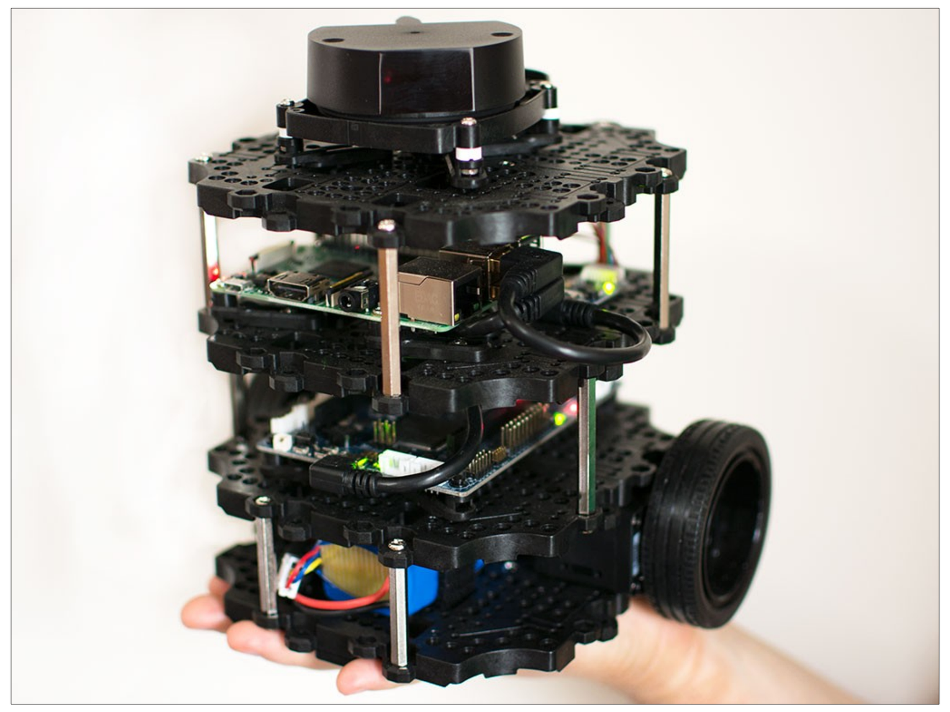
Conference or Workshop Item:
Foster, Simon David orcid.org/0000-0002-9889-9514 (2022) Verifying and Assuring Robotic Systems with Isabelle/UTP. In: YorRobots and RoboStar Industry Exhibition, 11-12 Oct 2022, University of York.

Simon Foster

Verifying and Assuring Robotic Systems with Isabelle/UTP



A hybrid automaton with differential equations in each mode



```
zstore robot_state =
  xcoord::ℤ  ycoord::ℤ — ‹ Robot position ›
  xvel::ℤ  yvel::ℤ — ‹ Robot velocity ›
  obstacles::"(ℤ × ℤ) list" — ‹ Obstacle register ›
  where — ‹ Invariant: The robot is never where an obsta
  "(xcoord, ycoord) ∉ set obstacles"

zoperation Drive =
  pre "(xcoord + xvel, ycoord + yvel) ∉ set obstacles"
  update "[xcoord′ = xcoord + xvel, ycoord′ = ycoord + yve

zoperation Idle =
  update "[xvel′ = 0, yvel′ = 0]"

lemma Move_correct: "Drive() preserves robot_state_inv" b
lemma Stop_correct: "Idle() preserves robot_state_inv" by
```

Verifying a robot model in the Z abstract machine notation

```
procedure reverse' "XS :: int list" over state =
"xs := XS; ys := [];
 while xs ≠ []
 inv length XS = length xs + length ys
   ∧ (∀ i<length ys. XS!i = ys ! (length ys - Suc i))
   ∧ (∀ i∈{length ys..<length XS}. XS!i = xs ! (i - length ys))
 do
    ys := hd xs # ys; xs := tl xs
 od"

lemma reverse'_correct: "H{True} reverse' XS {ys = rev XS}"
  — ‹ Generate Verification Conditions ›
  apply vcg
  — ‹ Discharge VCs using sledgehammer ›
  using hd_conv_nth less_Suc_eq apply fastforce
  apply (metis Suc_diff_Suc Suc_le_lessD list.exhaust_sel nth_Cons_Suc)
  apply (simp add: nth_equalityI rev_nth trace_class.less_iff)
  done
```

Verifying imperative code with Hoare logic and VCG

The Challenge

- Robotic Systems share variables with the real-word corresponding to **physical quantities**.
- **Challenging to verify** due to heterogeneous paradigms: reactive, concurrent, cyber-physical.
- We cannot compute every possible behaviour (state explosion), but only **approximate** them.
- Alternatively, we can model **all** behaviours by characterising them **symbolically**.
- We need a scalable tool that can handle their inherent **semantic heterogeneity**.

The Solution: Theorem Proving with Isabelle/HOL

- **Isabelle/HOL**: a state-of-the-art interactive theorem prover and assured development platform.
- Trustworthy by design (LCF), highly flexible, and **scalable** to large developments.
- High degree of **automation**, including SMT solvers, Computer Algebra Systems, etc.
- Allows us to model any mathematical concept symbolically, and prove theorems about it.
- Can model and verify CPSs, including a controller model, the differential equations, and code.

An autonomous boat

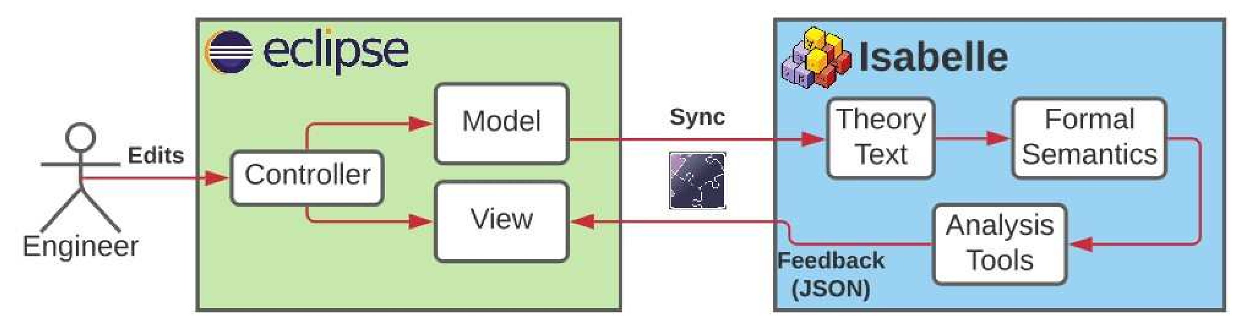
```
statemachine LRE_Beh =
uses LRE_State
vars v::"real[meter/second]" h::"real[radian]"
events reqMOM reqOCM reqHCM
  reqVel::"real[meter/second]"
  advVel::"real[meter/second]"
  reqHdng::"real[radian]" advHdng::"real[radian]"
states OCM MOM:"entry advVel!(1.0)"
  HCM:"entry advVel!(0.1)" CAM
initial OCM
transitions
  t1: "from OCM to MOM trigger reqMOM
       condition vel ≤ 0.1 ∧ odist(cdyn) > 7.5·meter
                ∧ odist(cstc) > 0.3·meter ∧ ¬ inOPEZ"
  t2: "from MOM to OCM condition inOPEZ"
  t3: "from MOM to OCM trigger reqOCM"
```

A RoboChart safety controller state machine

```
dataspace AMV =
  constants S::ℝ f_max::ℝ
  assumes fmax:"f_max ≥ 0"
  variables p::"ℝ vec[2]" v::"ℝ vec[2]"
    a::"ℝ vec[2]" ϕ::ℝ s::ℝ

abbreviation
  "ODE ≡ { p' = v, v' = a, a' = 0, ϕ' = ω }"

lemma "{s² = v · v} ODE {s² = v · v}"
  by (dWeaken, metis orient_vec_mag_n)

lemma "{a = 0 ∧ v = V} ODE {a = 0 ∧ v = V}"
  by (dInduct_mega)
```

Verifying the kinematics with differential induction

Can we model a complex autonomous transport network?



Integrating Isabelle into Model-based Engineering

Code generation via Haskell

Verifying Robotic Systems with Isabelle/UTP

- **Isabelle/UTP** is our library for verification tools based on *"Unifying Theories of Programming"*.
- Uses **heterogeneous semantics** for combining varied programming and modelling notations.
  - e.g. Z notation (ISO 13568), CSP, RoboChart, hybrid programs (used for modelling robots).
- Applied to creation of usable, efficient, and scalable **automated verification tools.**
- e.g. **differential induction**: verifying non-linear ODEs without needing explicit solutions.

Future Work

- Integrating Isabelle into development processes with **bidirectional model transformations**.
- Verifying complex concurrent robots (e.g. fleets, swarms) using **compositional verification.**
- **CyPhyCircus:** a semantically heterogeneous formal robotics modelling language.
- Animation and visualisation using code generation and the Functional Mockup Interface (FMI).
- **Verified code generation** for deployment on a physical robotic platform (e.g. ROS).