Testing Embedded Control Systems combining Hardware-in-the-loop Simulation and Temporal Logic

Marco Aurelio Antonio Sanvido

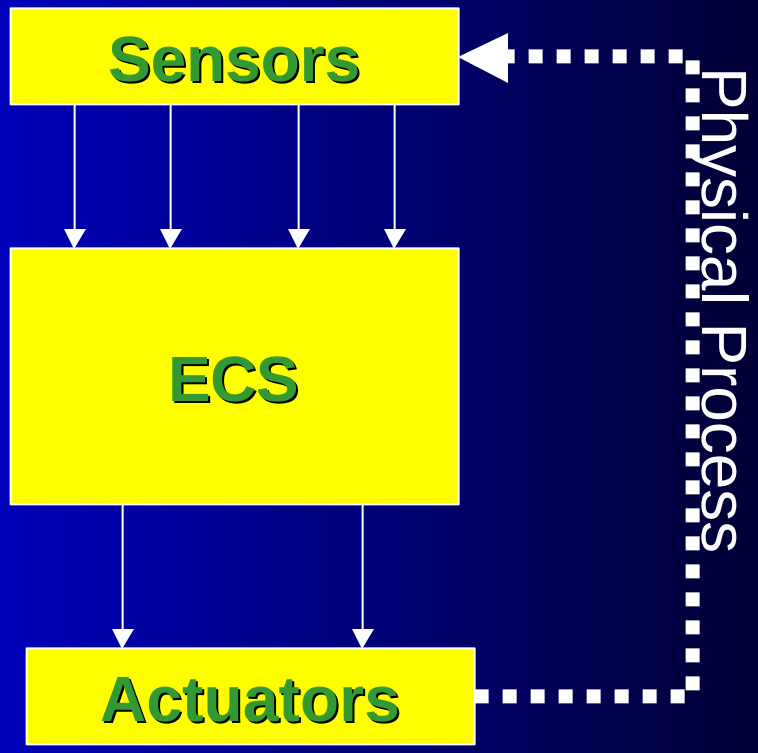
Outline

- Motivation: ECS Testing
  - ECS Properties
  - HIL Simulation for ECS
- HIL Simulation
- Fault Specification Language (Fausel)
  - Testing with Fausel
  - A small Example
- Conclusion & Some Demos

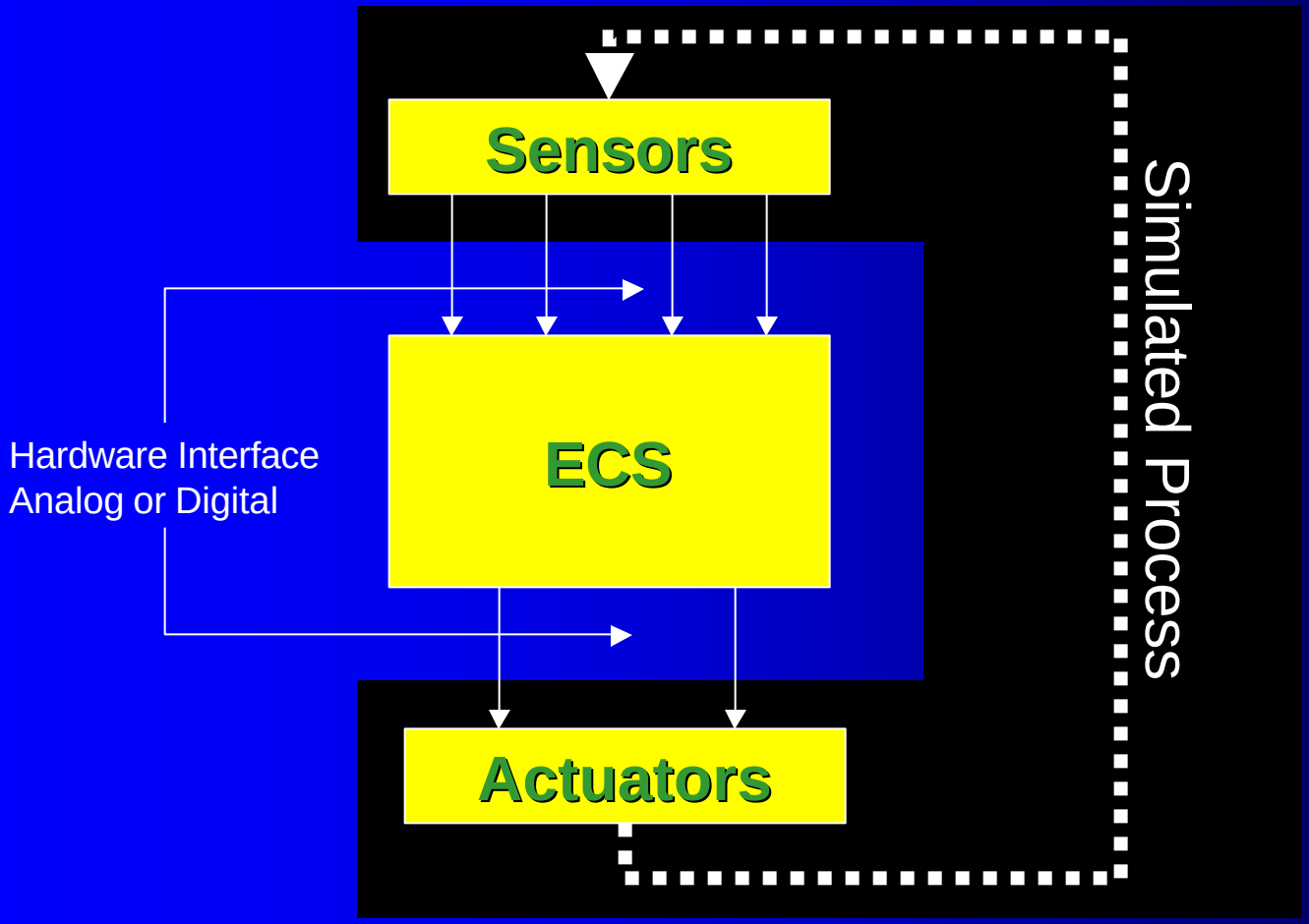
Embedded Control System

Unique Properties of ECS Software:

- Timeliness
- Concurrency
- Liveness
- Reactivity
- Safety



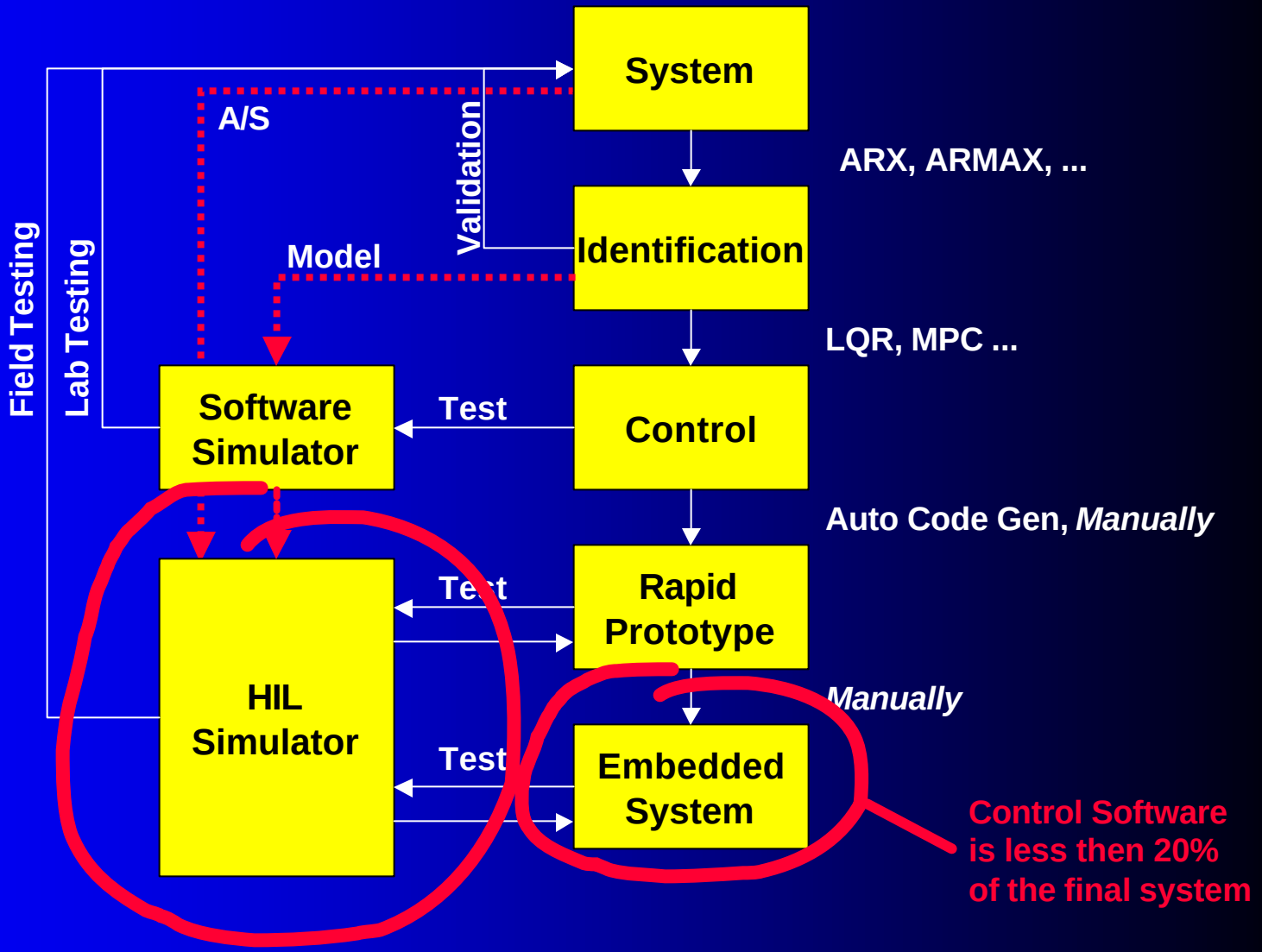
Hardware-in-the-loop Simulation



1. The simulated process can be operated with the *real* control hardware
2. The simulated process replaces either fully or partially the controlled process consisting of actuators, physical process and sensors.

(Isermann et al. ‘99)

ECS Development




OLGA helicopter crash

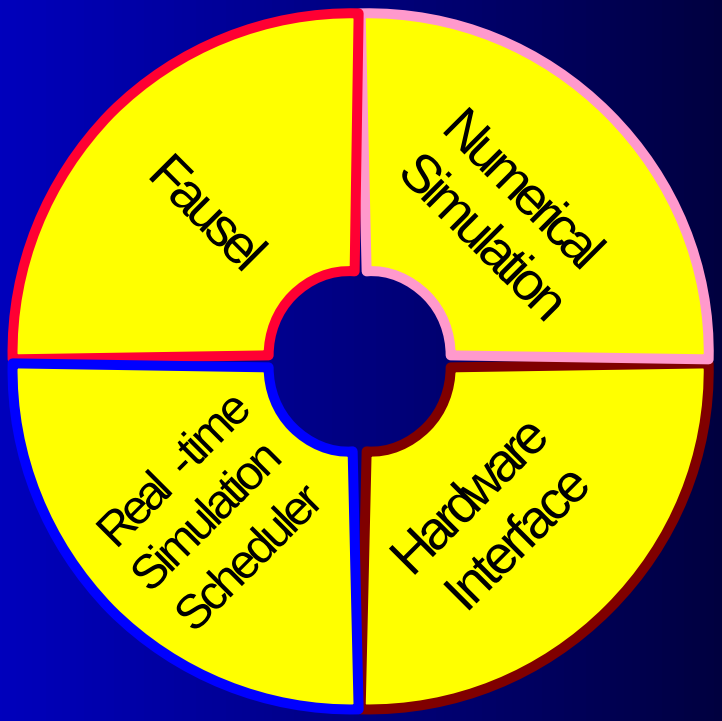
Hardware-in-the-loop Simulation

Why do we need HIL Simulation?

- Allows testing dangerous situations
- Ensure a *correct* implementation, reducing the gap between design and implementation
- Allows repeating the same test again and again deterministically

The HIL Simulation

- Simple implementation of the mathematical formulas of the simulated process.
- Generalized signal acquisition and generation.
- Real-time simulation scheduler.
- Generic fault generation/verification engine (Fausel).



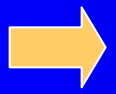
Fausel (FAUlt SpEcification Language)

The HIL generates fault sequences:

- Deterministic, non-deterministic, configurable
- Logically interconnected,
  e.g. Fault ***B*** can start only if Fault ***A*** has happened once

The HIL tests the ECS response specification:

- Tests for temporal behavior (using LTL)
- Tested during simulation time

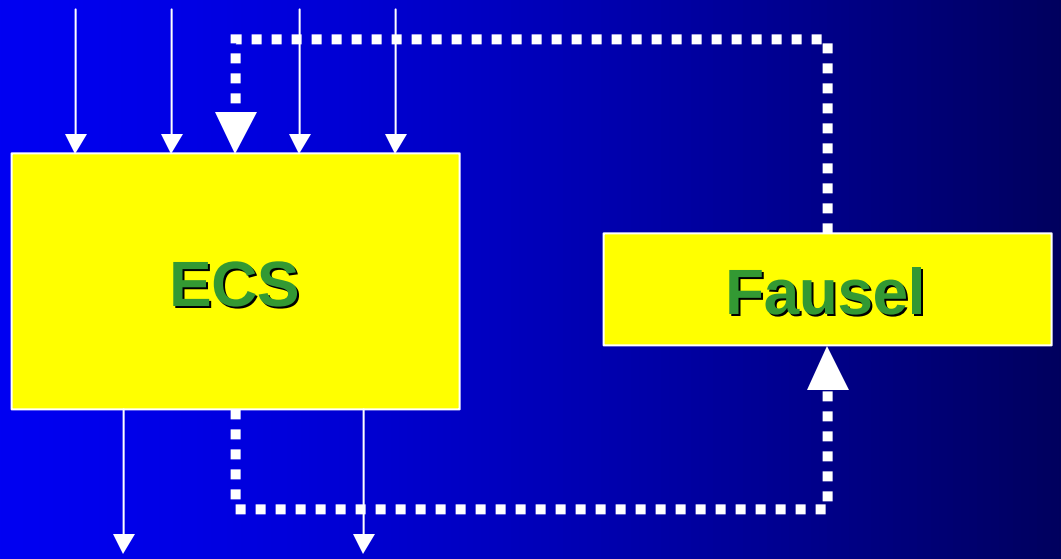


regression tests, documentation
on the real system are possible

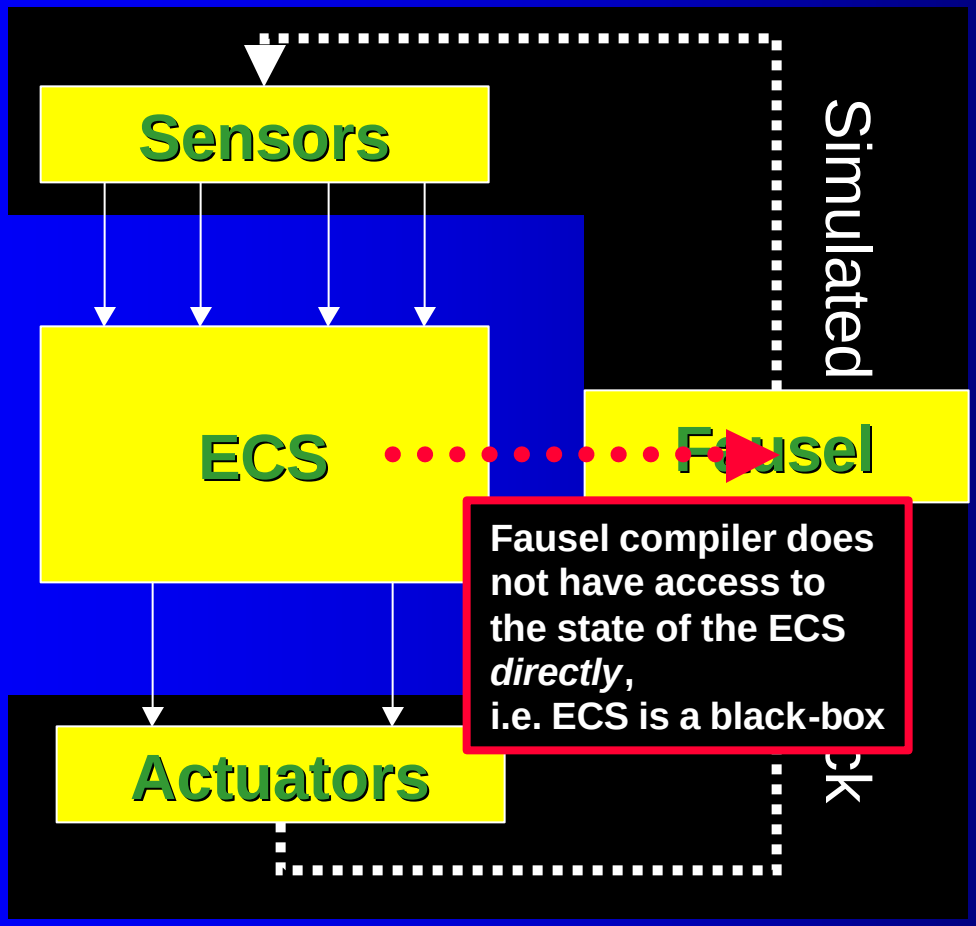
ECS Testing with Fausel

Fausel goals:

- ECS correctness ≠ checking the final answer
- ECS correctness = stream of partial answers



ECS Testing Integrated into the HIL




```
SIMULATE Helicopter;

  FAULT Wind0 IS Wind(w:=10.0)
  START (pitch > 0.1);
  STOP firsttime + 10.0;
  PERIOD 0.5;
  SPEC
    F[5,10](yaw = startyaw);
  END Wind0;

END Helicopter.
```

A Small Example of Fausel

How can I test that during an auto-piloted flight a wind gust is *correctly* handled?

Correctly could mean:

1. the quadratic yaw error ***E*** is in a tolerable value ***T*** between ***x*** and ***y*** seconds. $F_{[x,y]}(E<T)$
2. The system reaches the goal in ***x*** seconds and holds it. $F_{[0,x]}G(E=0)$
3. The error is never bigger than ***T*** between ***x*** and ***y*** seconds. $G[x,y](E<T)$

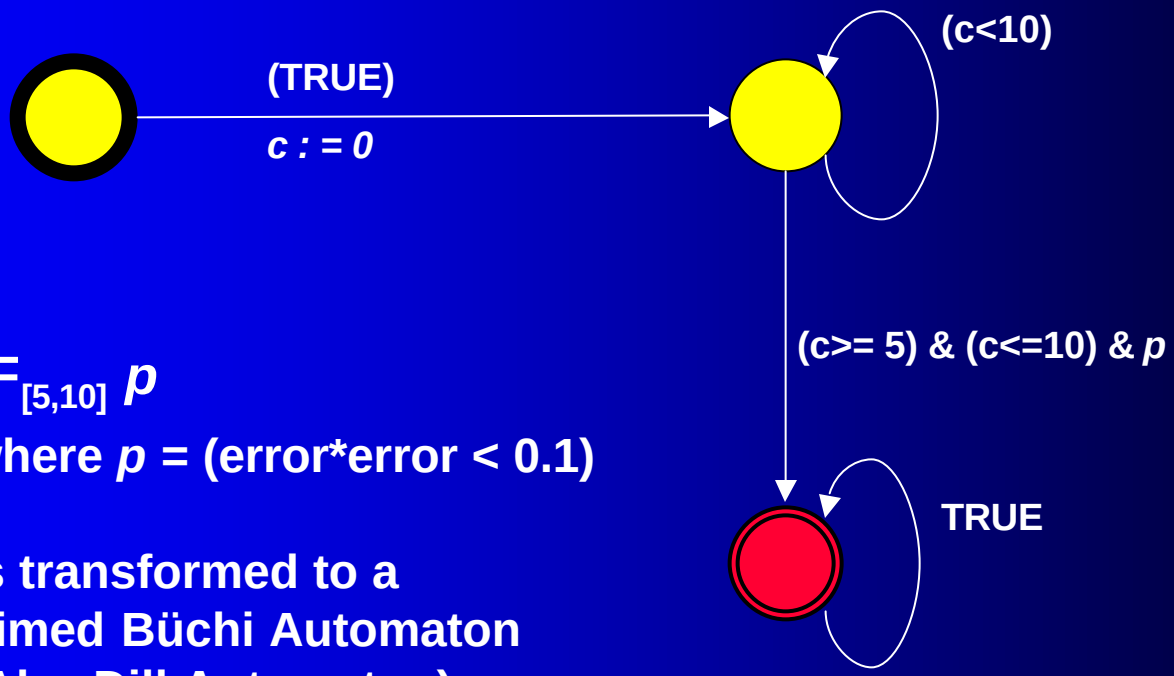
A Small Example of Fausel

```
SIMULATE Helicopter;

  FAULT W   Helis.WindGust(wind := 0.2; windtime := 2.0);
   START (pitch < -0.05);
  STOP firststart + 30.0;
  PERIOD 0.5;
  F[5,10]((yaw - startyaw)*(yaw - startyaw)< 0.1)

END Helicopter.
```

Example of a forward and backward maneuver, with error generation (and testing)

Play Simulation

A Small Example of Fausel

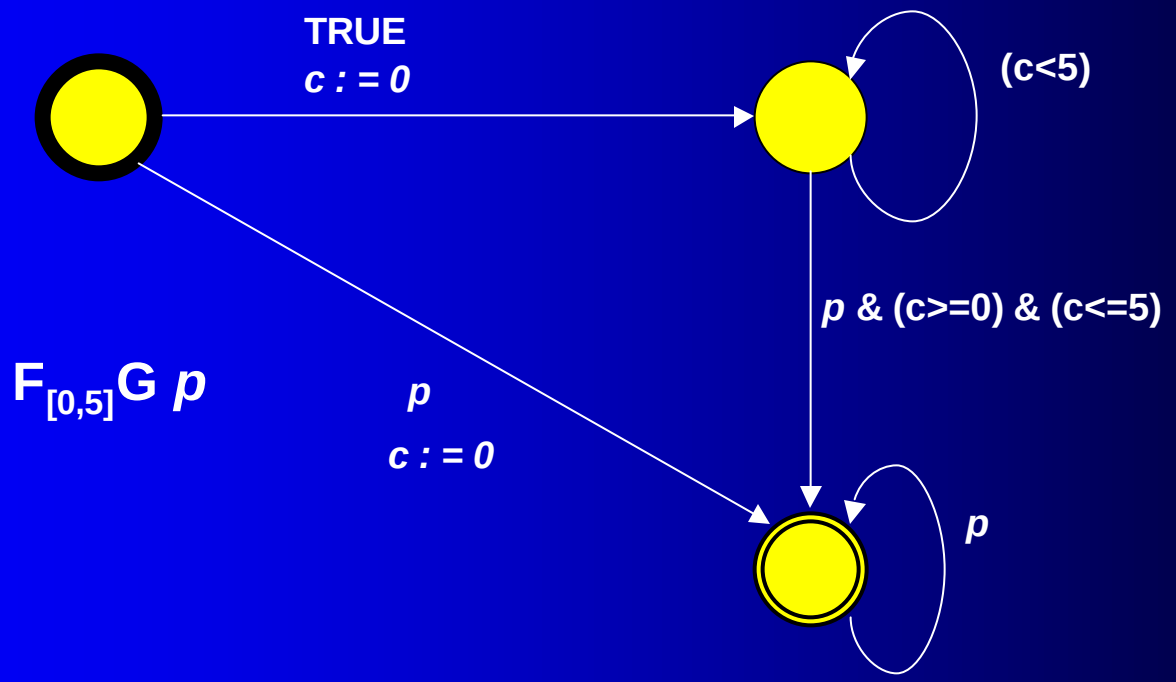


$F_{[5,10]}\ p$

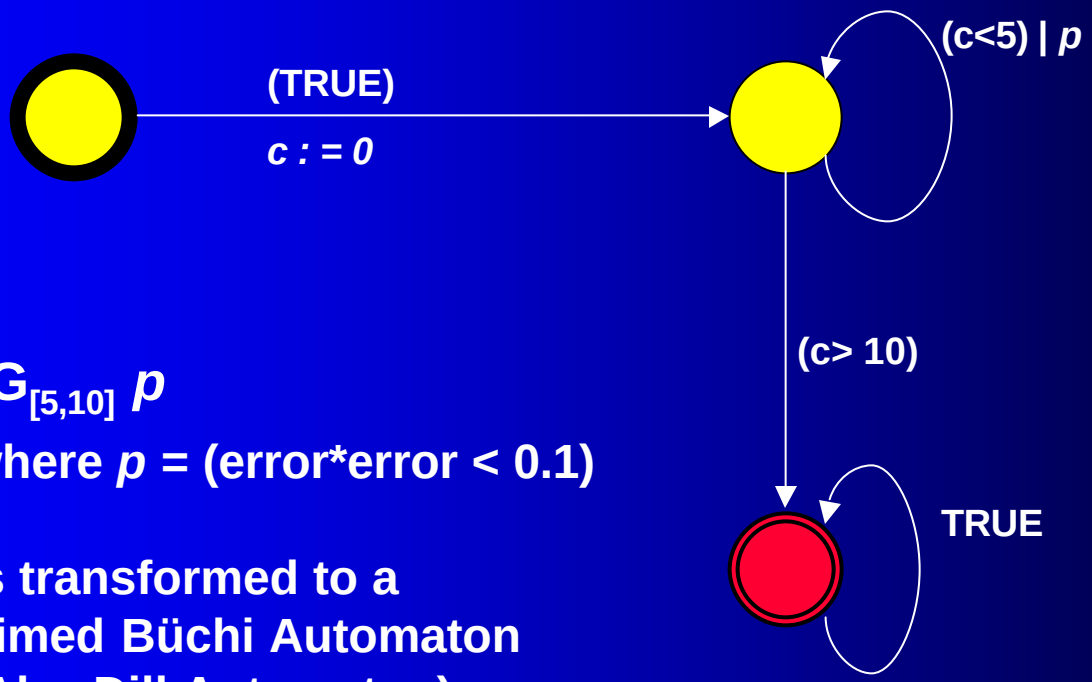
where p = (error*error < 0.1)

is transformed to a
Timed Büchi Automaton
(Alur-Dill Automaton)

A Small Example of Fausel



A Small Example of Fausel



$G_{[5,10]}\ p$

where p = (error*error < 0.1)

is transformed to a
Timed Büchi Automaton
(Alur-Dill Automaton)

Test Classes

What are we able to test?

1. We are only able to test expected faults!
2. We are only able to generate errors on the Model, Sensor and Actuators
3. We are NOT able to check ECS states (only possible via online-debug or extrapolation)
4. The *Temporal Logic* is *reasonably* expressive, but not complete. E.g. “*p* must be *TRUE* even times”
5. Test sequences are finite, but a specification could be infinite, e.g. G ***p*** => use constrained temporal logic (MTL)

Conclusions and Outlook

Has been applied to:

1. Barrage Simulation (Simplified Demo)
2. OLGA Helicopter Main Rotor Controller
3. OLGA Helicopter Hovering Controller (Video)

Framework Extensions:

1. Java implementation is available (Demo)
2. Matlab interface (Demo …)
3. Integrating Temporal Specification Testing into Junit (Open Project!!!)