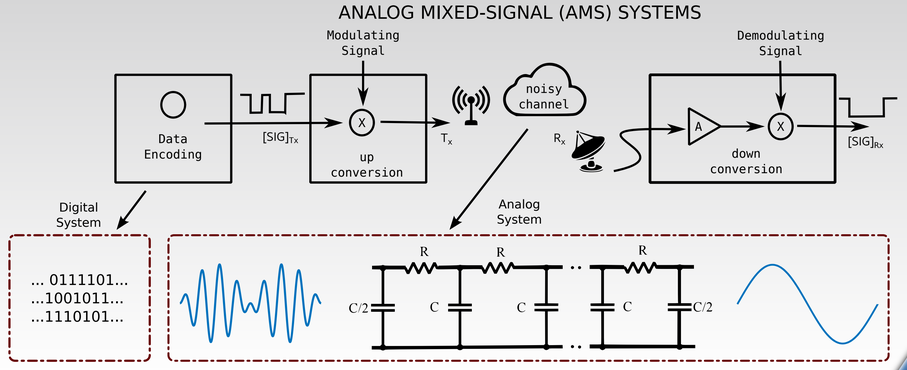
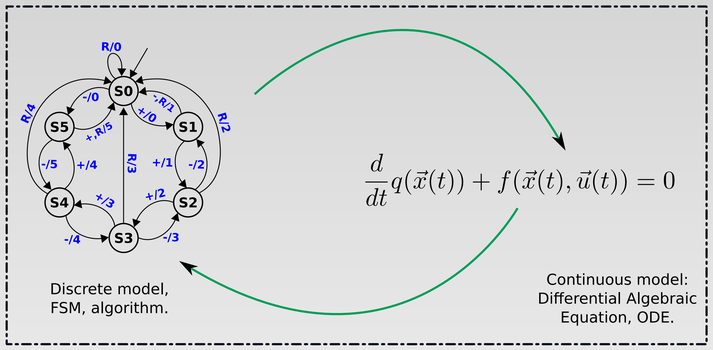
Accurate Booleanization of Continuous Dynamics for Cyber-Physical Systems

Archit Gupta
UC Berkeley

ANALOG MIXED-SIGNAL (AMS) SYSTEMS



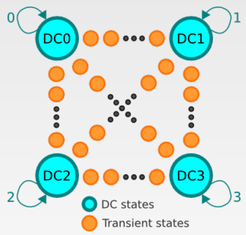


$$\frac{d}{dt}q(\vec{x}(t)) + f(\vec{x}(t), \vec{u}(t)) = 0$$

Discrete model, FSM, algorithm.

Continuous model: Differential Algebraic Equation, ODE.

- In both AMS and CPS, a discrete system interacts with a dynamical system.
- Our objective is to verify a formal property of the overall system, for example:
  - In case of the communication system, when do we get a bit error?
  - In case of a set of unmanned aerial vehicles, what can be the failures in a planned path or a proposed collision avoidance algorithm?
- Verifying such a property for a dynamical system, a.k.a reachability, is hard.

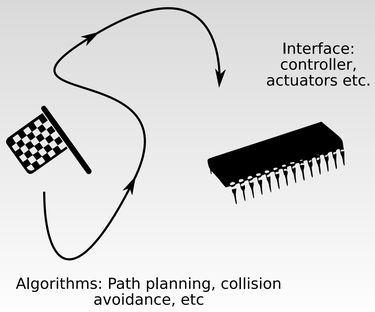


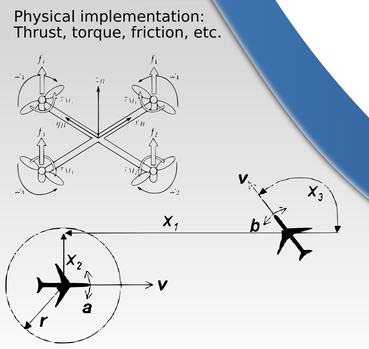
Approximate the continuous model with a discrete system

"Approximate the continuous system with a discrete one, and verify the overall discrete system!"

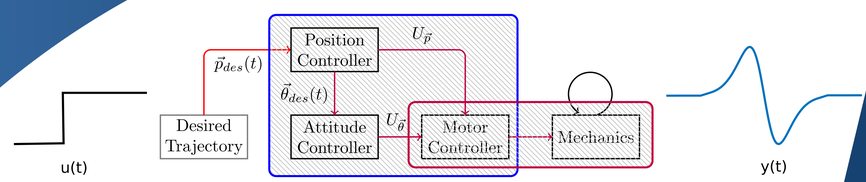
Theme of an ongoing project for AMS verification called ABCD-NL.

CYBER-PHYSICAL SYSTEMS





TRAINING



- For a predefined set of piecewise linear training inputs, system is simulated to obtain the output trajectories.
- Values of some internal unknowns that the user believes are relevant to the Input-Output behavior are also included.

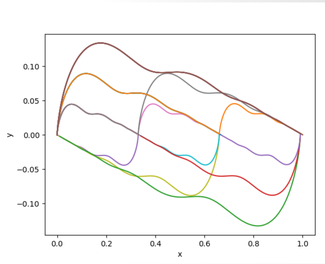
MODELING

Using both the open source MODSPEC API and Verilog-A for modeling the analog aspects.

```
module SpringMass(p, n);
// Verilog-A model for an autonomous Spring-Mass system
//       p. -----/\/\/\/\-----0------ .n
//                          ^^^ Internal Node (i)
// The spring is ideal, so the net force on it should be zero.
        electrical p, n, i;
        parameter real sp__k=1k from (0:inf);
        parameter real mass=1m from (0:inf);
        branch (p, i) spring;
        branch (i, n) bob;
        real v__bob;
        analog begin
                I(spring) <+ sp__k * V(spring);
                v__bob = ddt(V(bob));
                I(bob) <+ ddt(mass*v__bob);
        end
endmodule
```

STATE-SPACE COVERAGE

State-space traversal for the prescribed training inputs.

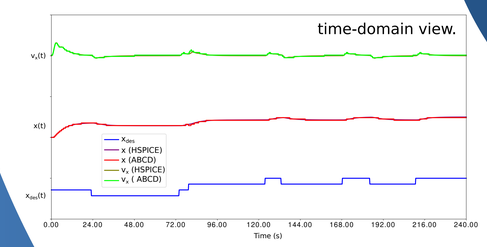


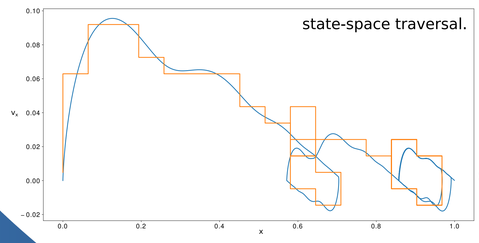
k^{th} trajectory corresponds to a solution of the DAE:

$$\frac{d}{dt}q(\vec{x}(t)) + f(\vec{x}(t), u_k) = 0$$

Solution for any initial condition x_0 on the trajectory remains on it as long as the input is held at u_k.

Algorithm: Follow trajectory t_k as long as input is held at u_k. When the input jumps to $u_{k'}$, find a point of the trajectory $t_{k'}$ which is close to the current one in the state space and JUMP.

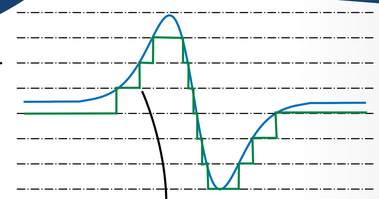




Some comparisons between the ABCD-generated FSM and the original dynamical system for randomly generated inputs.

VERIFICATION

TRANSLATION



1. Discretization,
2. Transition-tables,
3. Jump Heurisitics.

1. ***Discretize*** the output signal using predefined (user supplied) discretization levels.
2. ***Record time(s)*** of transitions across the discrete boundary values.
3. Extract the **JUMP**(s) at each discrete state for all possible input values.

PhD Advisor: Jaijeet Roychowdhury.
Poster template by Felix Breuer.