# Notes S5 breakout session - Hybrid Automata Verification
# S5 Conference – June 2015

**Introduction**

- What is the definition of nondeterminism we are considering? Certification nondeterminism? Usually there is a distinction between internal vs external nondeterminism.

- Does a system with the same input produce the same output? No, we need to provide some way to resolve nondeterminism in resets (discrete post), and differential equations (continuous post)

How is invariant generation done?

- templates and test, sum of squares programming, manual reasoning

- Is this related property-directed reachability? Not sure, haven't seen this for hybrid automata yet.

- automating simulink to do invariant generation, UVC (Dr. Hu?) parses C code to extract invariants

- Simulink block approach, manually give invariants

- Hynger tool (ICCPS 2015) for invariant generation from Simulink diagrams using Daikon

**Stanley Bak**

- AFRL-RI. Why was hycreate created rather than using other tools? It was made before Flow* so there was limited tool support for nondeterminism. Also, no tools had implemented mixed face lifting.

**Ryan Turner**

- AFRL-RI. Emergent behavior research, using hybrid systems tools to address multi-agent autonomy

**John Schierman**

- Barron Associates. background is in controls, interested to investigate what's different about hybrid systems

**Mike Devore**

**-** Barron associates. RTA design, interested in looking at how certification is approached. A lot of the proofs don't involve the set of states the system will get into, not just good states and bad states. Other properties are of interest, like functions over system states. Maximum frequency... optimization over hybrid automata models. Input constraints are not just bounded, but instead frequency domain properties. Frequency drops off after a certain bound. Bad states over a history of times.

- history of signals as the spec (not out of bounds for too long)
- Use monitor automaton (manually made), use binary search for optimization
- signal temporal logic
- statistical properties
- formalizing a monitor (FFT)

**Christ Elliott**

- Lockheed. Interested in properties in the frequency domain
- quantum v&V methods

**Lucas Wagner**

- Rockwell. Verfify software with model checking
- verified triplex voting, but took lots of effort. Challenge was in the switch from three sensors to two sensors, smoothing out transients. Took 1-2 years of effort was linear
- is that something could be verify quickly using hybrid automata tools? it would be useful if so.
- The model was software was component, so we used software model checking techniques (hybrid automata don't yet generally handle software as a part of the model)
- hybrid dynamic logic might be able to support software better.

**Pavithra Prabhakar**

- Kansas State University. developer of tools, one challenge is the number of parameters you need to set

- comparing tools is complicated because of the parameters

**Sagar Chaki**

- SEI. model checking and software verification LTL Kripke structures

- verification of distributed systems, want to prove collision avoidance, but only can prove properties about software

- want to verify using multiple types of analysis SW -> model checking, physical -> hybrid systems tools. Combined using assume guarantee reasoning

- engineers have some idea of why it works; this should be analogous to the verification

- competitions are very useful for the verification community. originaly sat solving, then hardware model checking, now software model checking. need standard input output format (AIGER for hw, for sw they use form of C code), and clear set of rules so results are not contested. Properties are assertions or reachability. Challenges are transparency, make sure environment is good. The quality of benchmarks is key, need industry support from the start. For example, in SW competition there are categories of programs (recursion, dynamic memory)

**Mauricio Castillo-Effen**

- GE research, sees hybrid systems every day. Aviation mining, oil and gas, power. Typical process is extensive simulation. Think better tools are possible.

- tool development is often separate from the real problems, need to bridge to the problem. Other tools may be helpful. Patterns to help bridge the gap. A taxonomy of properties is needed. Developers don't typically say they have a reachability problem.

- work on how to take a problem, and convert to another from that the tools can handle

- microsoft SLAM project is a success. verification of device drivers by introducing asserts. Now it's part of the standard development process. Need to agree on type of constraint for HA.

**Tim Arnett**

- Student, university of cincinatii in aerospace. Using a controller represented as a hybrid automaton, and want to do verificaiton

- Control goal is to stay alive as long as possible

**Matt Clark**

- AFRL-RQ. Rule-based systems are hybrid systems, but the conversion hasn't really been done before. Fuzzy systems can have stability proven stable using hybrid systems tools. Frequency tracking in fuzzy set may be a challenge. The controller tracks really well up to cutoff frequency, and then it cuts off very sharply, not graceful degradation. Relating to bandwidth and robustness. One could trade off response for graceful degradation (robustness).

**Bob Grabowski**

- MITRE. People often build robots without much testing and evaluation

- A direction could be to work with test community that wants to simulate and find failures

- from control theory to decisions. Need to find low probability events. There is a gap between these two parts

- Visualization of tools is important. HyCreate has a visualization during the computation

- nondeterminsism and sensitivity, how close is the system from becoming unsafe?

- testing is sparse, so manually try to find failure boundary. They don't have the reasoning for why something broke.  Diagnoisis is important, find out exactly where the problem was.

- with federated architecture testing, this comes up a lot. An integrated system is tested, but then assumptions become violated. Black box tools are useful. People will put pieces in so that the tests are passed.

- processed visualization can also be important, instead of just projections on variables. For example, the amount of time spent on different parts of the computation.

-  classes of autonomous systems to help standardized testing, standard checks. Tell the proprietary models to expose a set of "testing signals" which can then be used by the testing, so it's not just black box

**Aaron Fifarek**

- AFRL/Linquest. Used spaceex, used dreal/dreach, hsolver

- understanding of formalism needs to be better when starting out with tools

- a lot of background is needed to use these tools. Can engineered properties be used in the tools, without having to convert the problem to a form specific for the tool?

- requirement specification is a key problem

- How about verification engineers? This might be dangerous because each company has internal tools. We need general set of open tools that the certification authorities would accept.

- open tools are good. Is the future more towards general tools that everyone uses, and the IP is the product. Mathworks Design verifier is expensive... but they still need updates. They are popular because they give it away while people are in schools. User community is important.

- but engineers are more comfortable using things like design verifier, maybe because Matlab is familiar

- compositional verification is less mature than component-based verification