A Hybrid Architecture for Correct-by-Construction Hybrid Planning and Control

Smart Information Flow Technologies
Robert P. Goldman, Daniel Bryce,
Michael J.S. Pelican, and David J. Musliner
Carnegie Mellon University
Kyungmin Bae

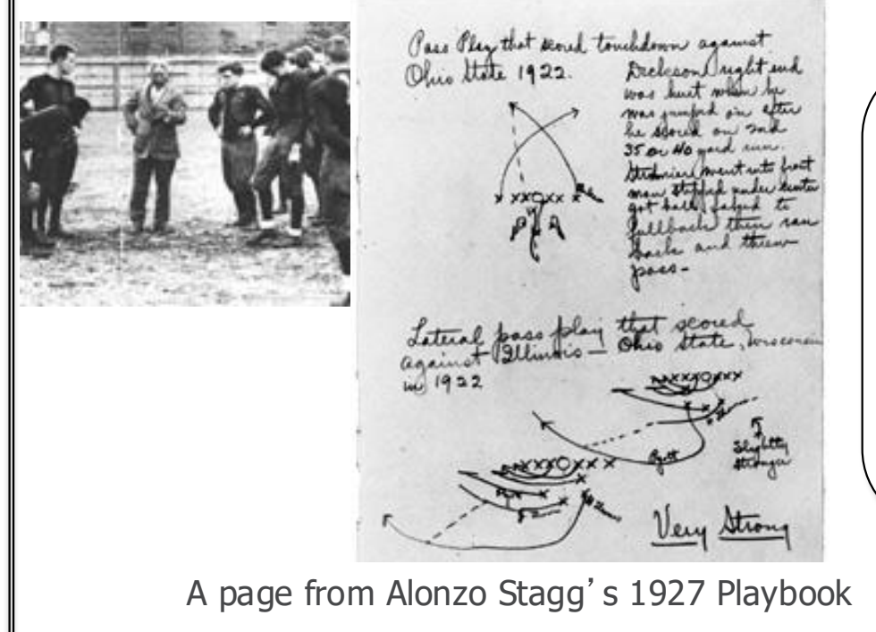
Originally presented at NASA Formal Methods Workshop,

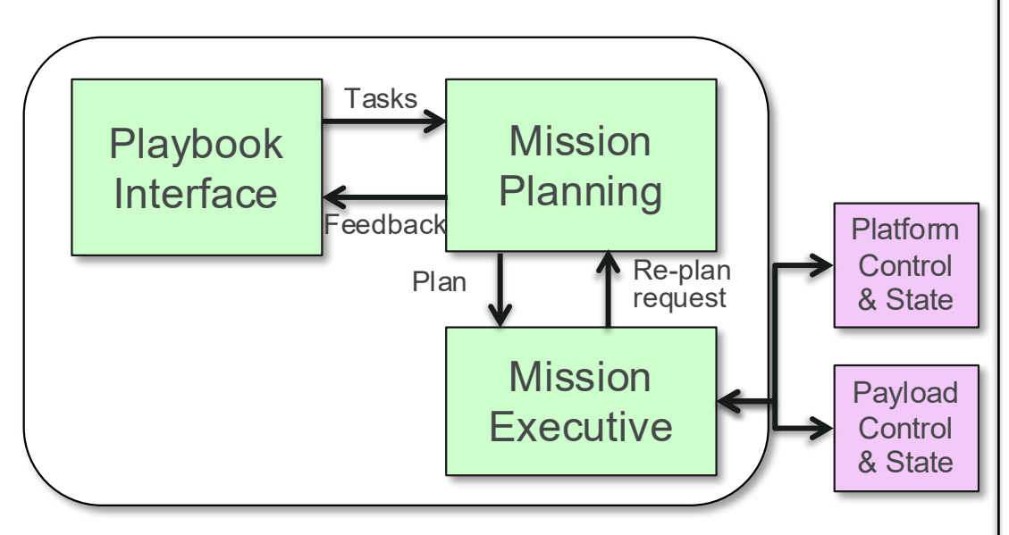
This research sponsored by the USAF under contracts FA9550-15-C-0030 and FA8650-16-C-2611. Views expressed here are solely the authors' and not those of the US government, USAF, or AFRL.

Goals

Big Picture: Effectively, reliably, and safely task teams of autonomous cyber-physical systems. Use formal techniques to ensure correct team control.
Specific: Extend previous tasking and control efforts to cover complex hybrid dynamics. Combine:
– SIFT's Playbook™ tasking interfaces.
– SIFT's D-CIRCA correct-by-construction distributed controller synthesis.
– CMU's dReach/dReal tool for non-linear hybrid systems verification.
Method:
– Layered planning/control of heterogeneous algorithms with downward refinement.
– Use *successive abstractions* to overcome efficiency challenges.
This Project: Proof of concept (Phase I) study of approach – Phase II implementation now underway.

Distributed Adaptive Planning and Control: Playbook

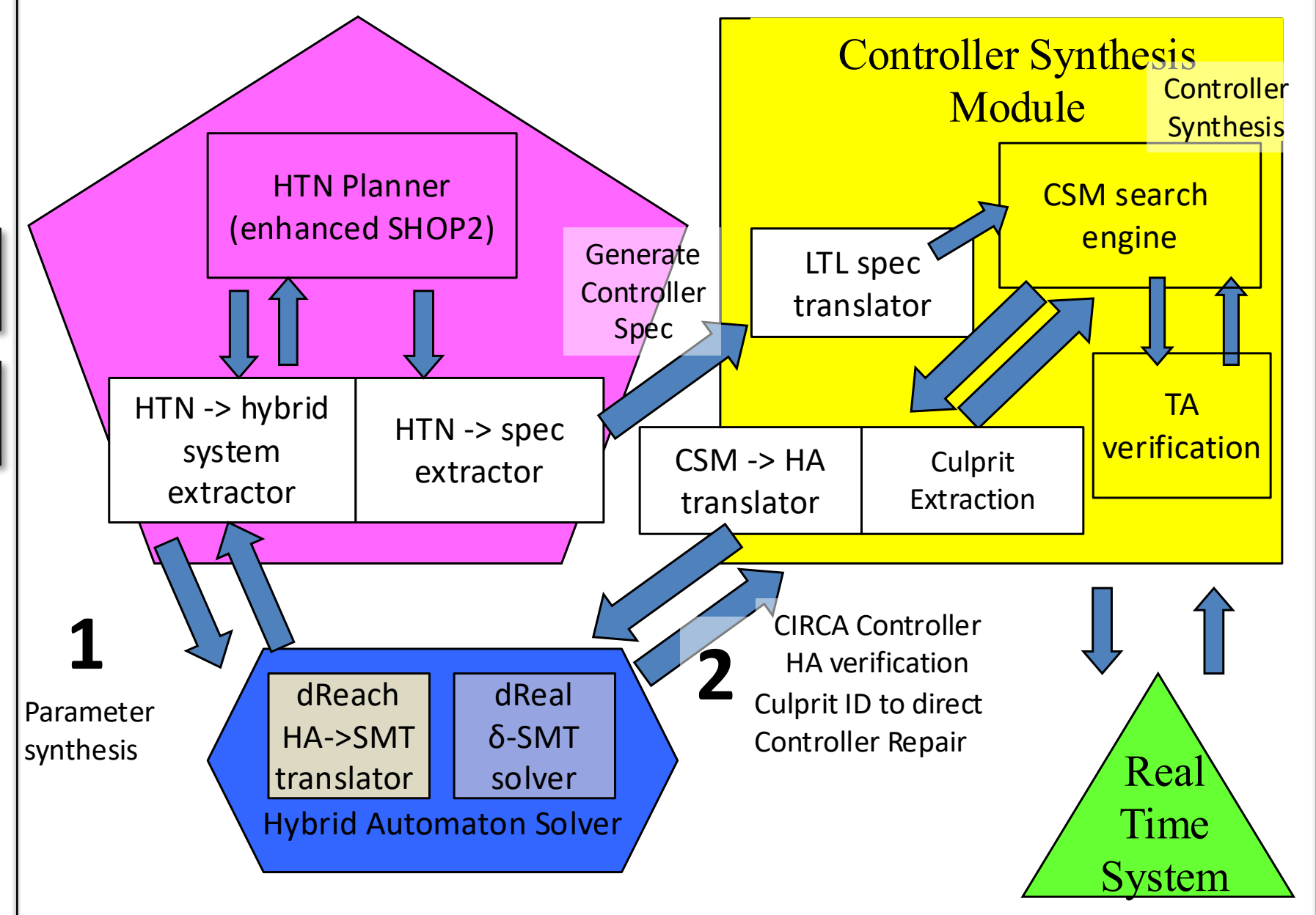

A page from Alonzo Stagg's 1927 Playbook



SIFT's Playbook Control Architecture

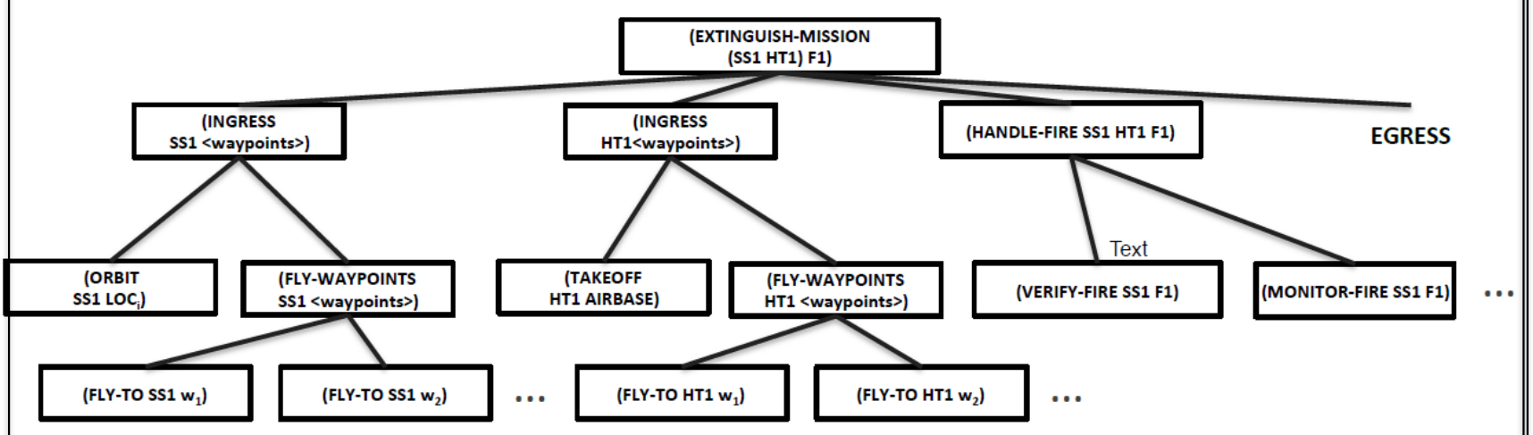
- ◆ SIFT's *Playbook*: Revolves around the concept of "*calling a play*"
  - Play "audibles" efficiently capture supervisor's intent.
  - Decision aids fill in details.
  - Plans can contain sub-goals that are assigned to lower-level units for planning and control.
- ◆ Playbook's hierarchical abstraction supports modular incorporation of new system capabilities via new plays.
- ◆ Supports supervisory control that scales to very large numbers of heterogeneous assets.
- ◆ Experimentally validated for UAV operators.

Hy-CIRCA System Architecture



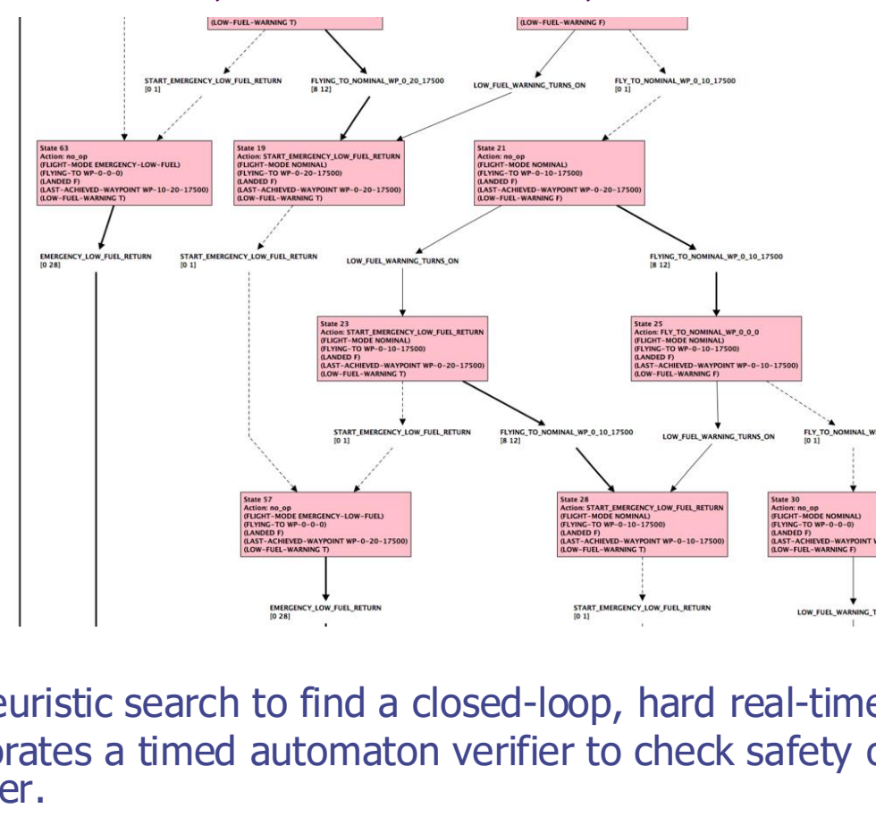
Component Technologies:

SHOP2: Mission Planner [Nau, *et al.*]



- ◆ Hierarchical task network planner.
  - Decomposes *tasks* into *subtasks* using *methods*.
  - Manages task-task, task-agent, task-environment, and resource constraints.
- ◆ Algorithm is sound and complete.
- ◆ Time-tested: 15+ year old software package.
- ◆ Open-source software library, with permissive license.
- ◆ Developed by University of Maryland; maintained by SIFT.
- ◆ Won multiple awards in the International Planning Competition.

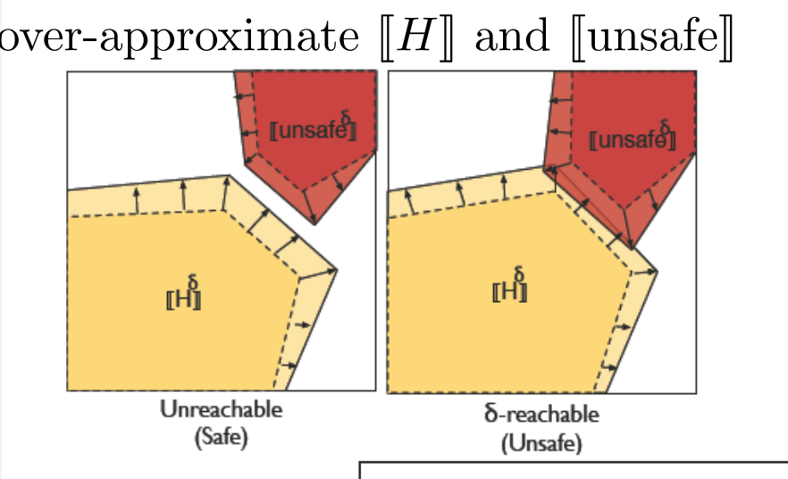
CIRCA Controller Synthesis Module [Musliner,Goldman,Pelican,*et al.*]



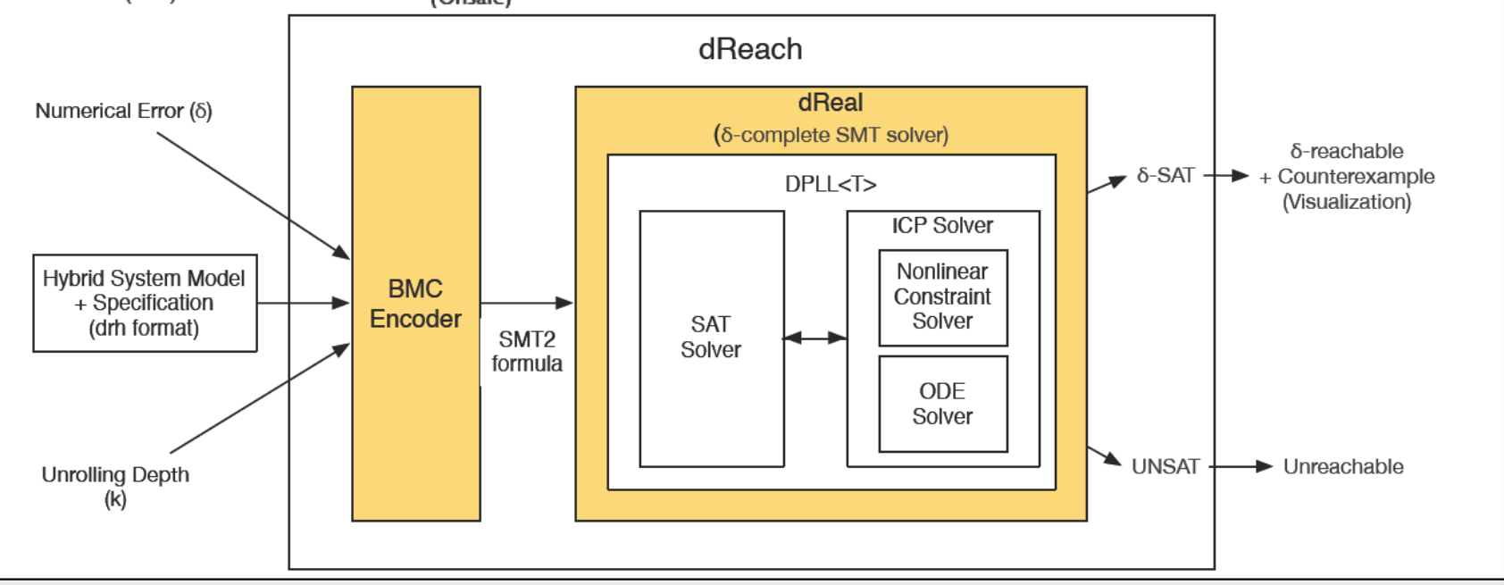
- ◆ Uses heuristic search to find a closed-loop, hard real-time discrete controller.
- ◆ Incorporates a timed automaton verifier to check safety of synthesized controller.
- ◆ Verification failures trigger counter-example guided backtracking to revise the controller.
- ◆ Timed automaton model is conservative approximation of controlled and uncontrolled continuous processes:
  - ◆ Consider worst case execution time of controlled processes.
  - ◆ Consider full range of durations of uncontrolled processes.

δ-Reachability Tools: dReal and dReach [Gao & Clarke]

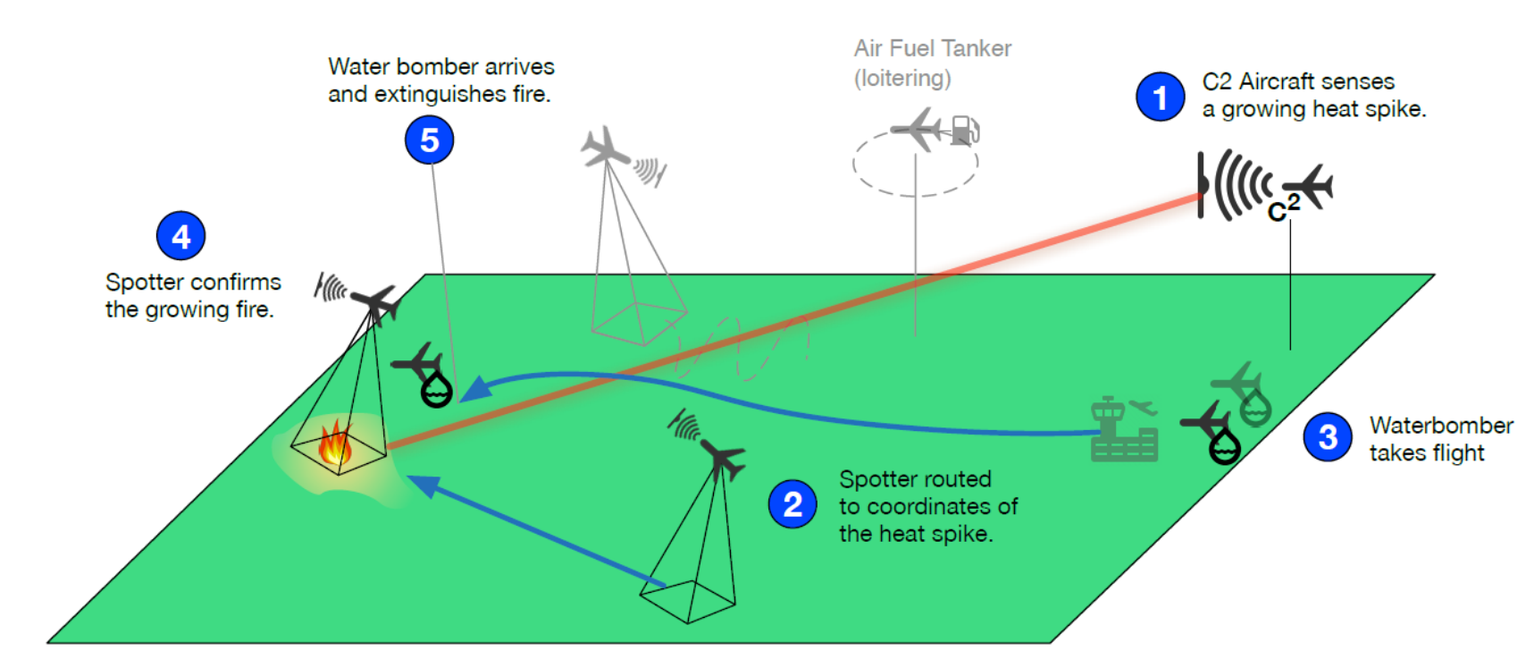
Given $\delta \in \mathbb{Q}^+$, $[\![H^\delta]\!]$ and $[\![\text{unsafe}^\delta]\!]$ over-approximate $[\![H]\!]$ and $[\![\text{unsafe}]\!]$



- δ-Reachability is decidable for a wide range of non-linear hybrid systems.
- δ-Reachability offers reasonable complexity bounds (PSPACE-complete).

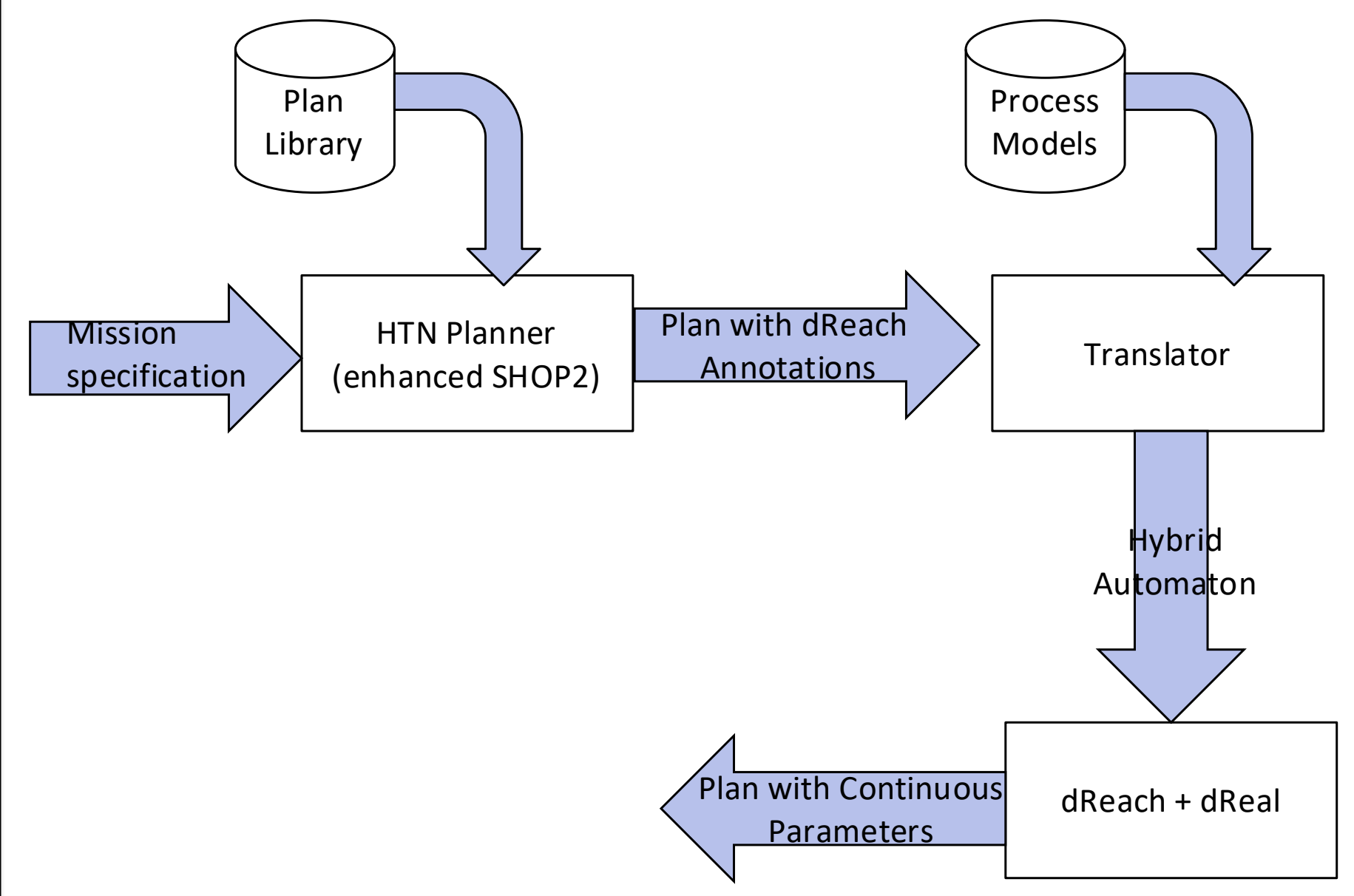

dReach
Numerical Error (δ)
Hybrid System Model + Specification (drh format)
Unrolling Depth (k)
BMC Encoder
SMT2 formula
dReal (δ-complete SMT solver)
DPLL<T>
SAT Solver
ICP Solver
Nonlinear Constraint Solver
ODE Solver
δ-SAT
δ-reachable + Counterexample (Visualization)
UNSAT
Unreachable

Test Problem: UAV Team Firefighting


Water bomber arrives and extinguishes fire.
Air Fuel Tanker (loitering)
1 C2 Aircraft senses a growing heat spike.
4 Spotter confirms the growing fire.
5
2 Spotter routed to coordinates of the heat spike.
3 Waterbomber takes flight

Mission plan involves addressing a fire by bringing a waterbomber, suitably loaded, and an orbiting spotter to the fire at the same time. The waterbomber will wait for targeting info from the spotter, then drop retardant.

Adding Hybrid Reasoning (parameter synthesis) to Mission Plan


Plan Library
Process Models
Mission specification
HTN Planner (enhanced SHOP2)
Plan with dReach Annotations
Translator
Hybrid Automaton
dReach + dReal
Plan with Continuous Parameters

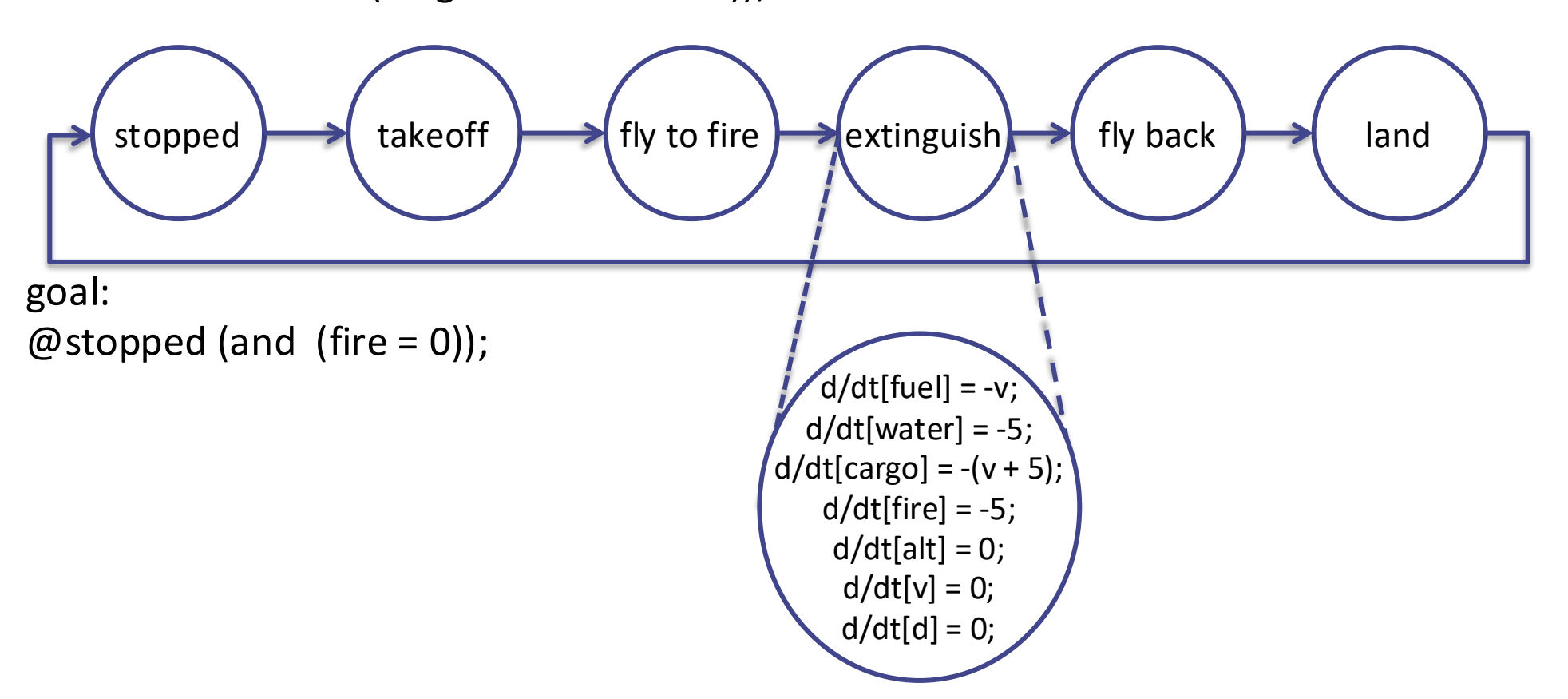
- ◆ Enhanced SHOP2
  - Uses forward reasoning to find a feasible plan (line through the state space);
  - Plan is conservative under-approximation (sound but incomplete WRT hybrid model).
  - As a side-effect, SHOP2 writes modeling information for dReach/dReal.
- ◆ Translator takes modeling information and database of continuous process information and composes hybrid model.
- ◆ dReal solves the resulting model to
  - Synthesize parameters for continuous processes and
  - Expands single line through state space to multi-dimensional tube.

Mission Plan

- ◆ Assign resources
  - Assign tanker to fire
  - Assign spotter to fire
- ◆ Bring assets to fire
  - Tanker ingress
    - Tanker takeoff
    - Tanker cruise
  - Spotter ingress
- ◆ Fire kill chain:
  - Verify fire
  - Monitor fire
  - Extinguish fire
  - Assess fire
- ◆ Return tanker to base
- ◆ Spotter perform recon

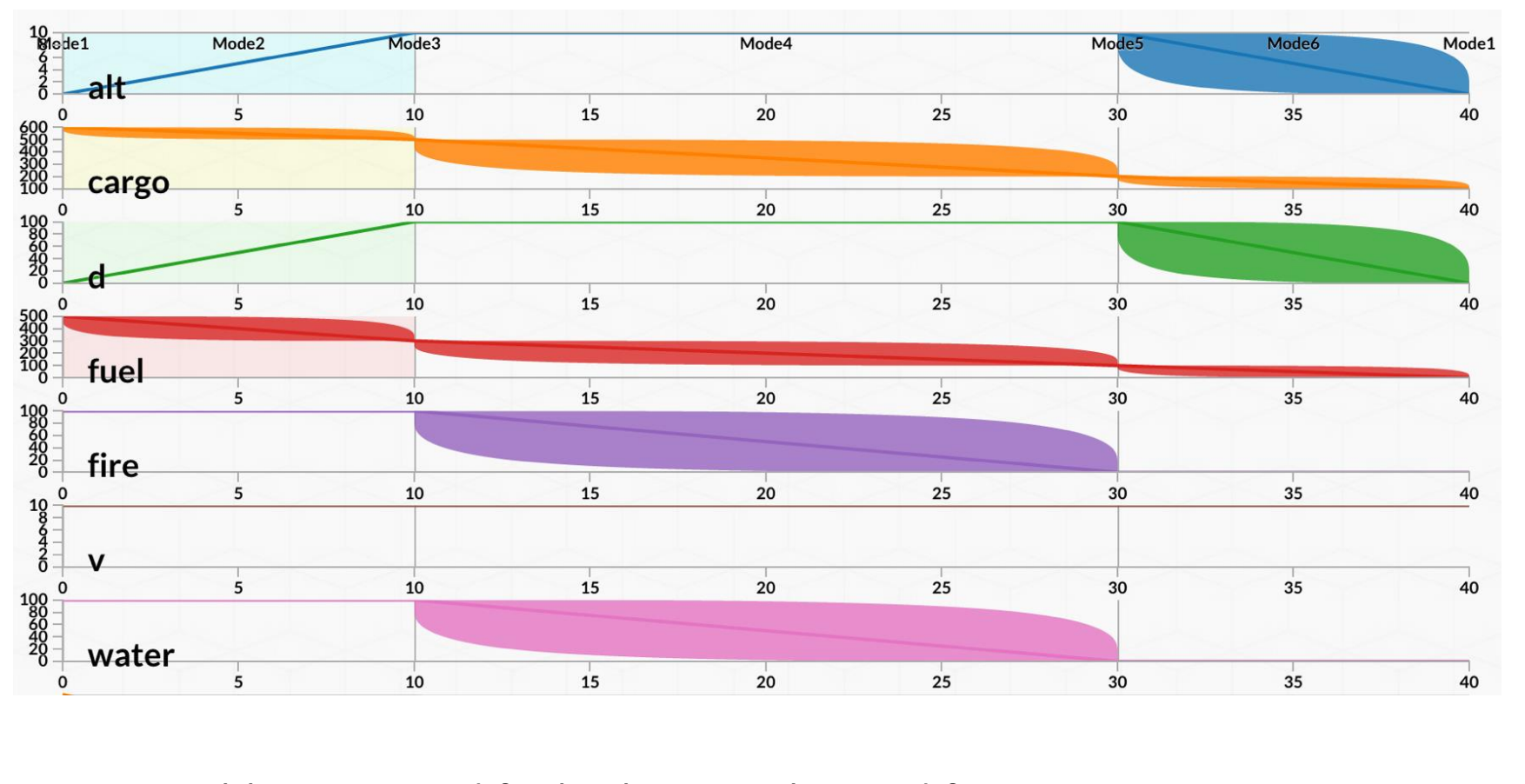
Hybrid Automaton for Parameter Synthesis

```
Initial conditions:
@stopped      (and (d = 0) (v = 0) (alt = 0)
                   (fire = 100)
                   (cargo = fuel + water));
```



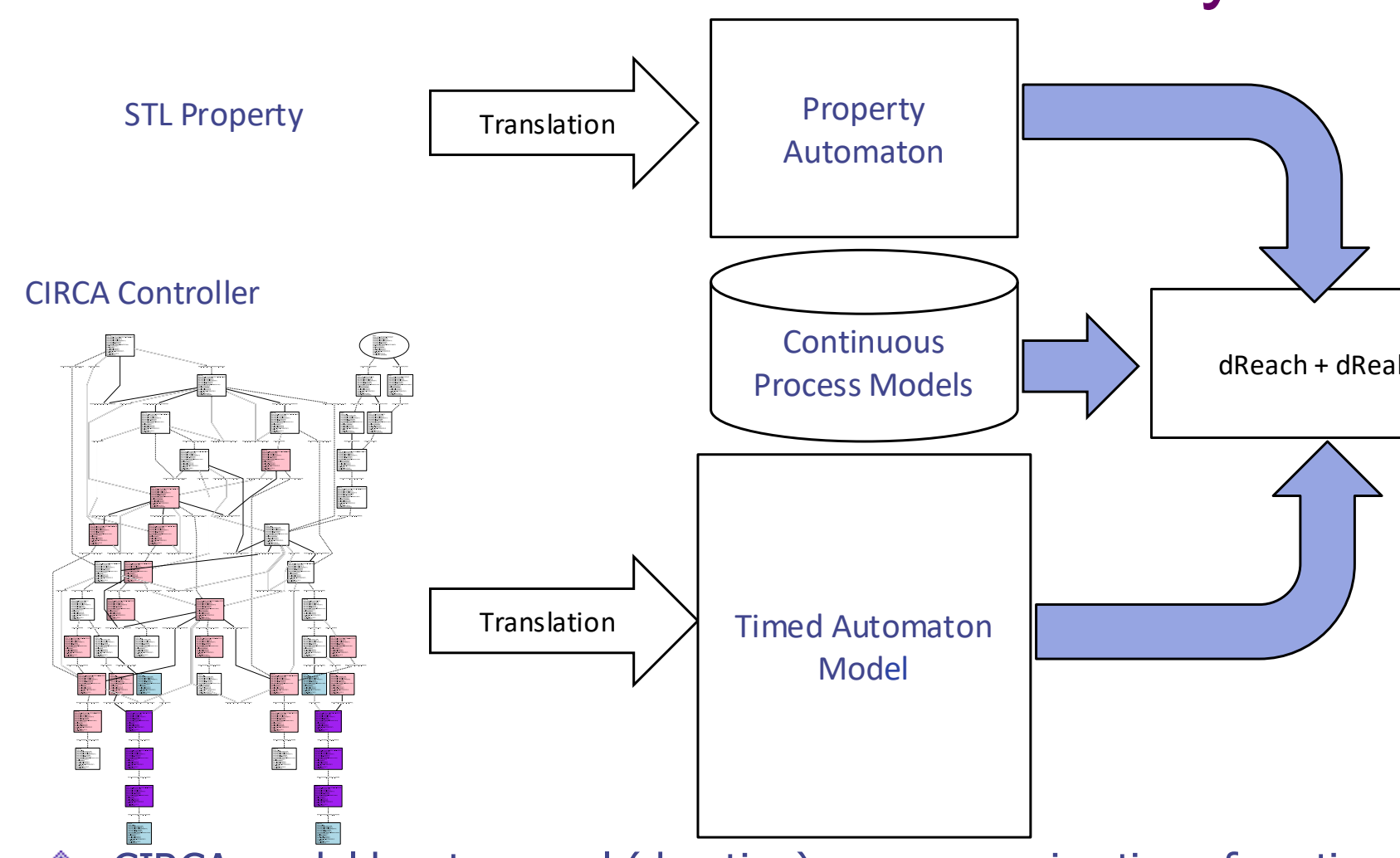
```
goal:
@stopped (and  (fire = 0));
```

Results of Parameter Synthesis



- Feasible water and fuel values synthesized from constraints.
- Involves multiple flight modes.

Hybrid Controller Verification



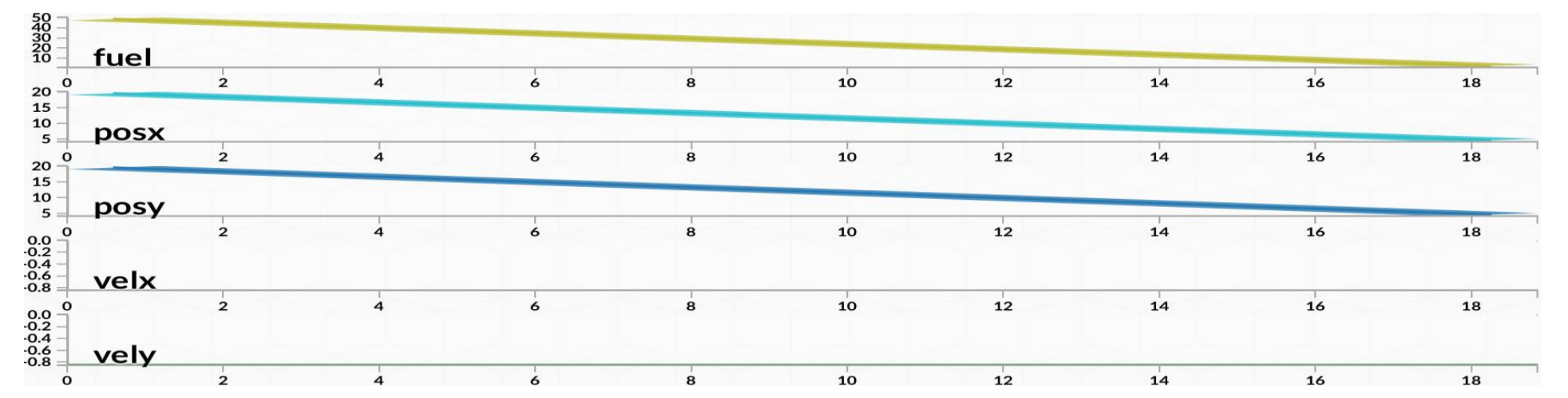
- ◆ CIRCA model has temporal (duration) over-approximation of continuous dynamics.
- ◆ dReach verifies STL claims using higher-fidelity models.
- ◆ If dReach fails to verify STL claim, CSM backtracks to repair the controller.

◆ Property: $\neg fuel\_empty\ \mathbf{U}_{[0,1000]}\ land$

◆ Continuous model

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{f} \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ -c_{eco} * (\sqrt{v_x^2 + v_y^2} + 0.1) \end{pmatrix}$$

◆ Examples:
- Initial fuel = 200: no failure
- Initial fuel = 50: counterexample



Conclusions

- Architecture for correct-by-construction plan/control synthesis.
- Hy-CIRCA uses successive refinement to tame complexity:
  - Mission planner reasons about resources over mission in open-loop.
  - dReal does nonlinear parameter synthesis in context of linear plan.
  - CSM does real-time, closed-loop controller synthesis with approximate model.
  - dReach only checks dynamics when controller is temporally correct.
- Now that feasibility study is done, working on full implementation.

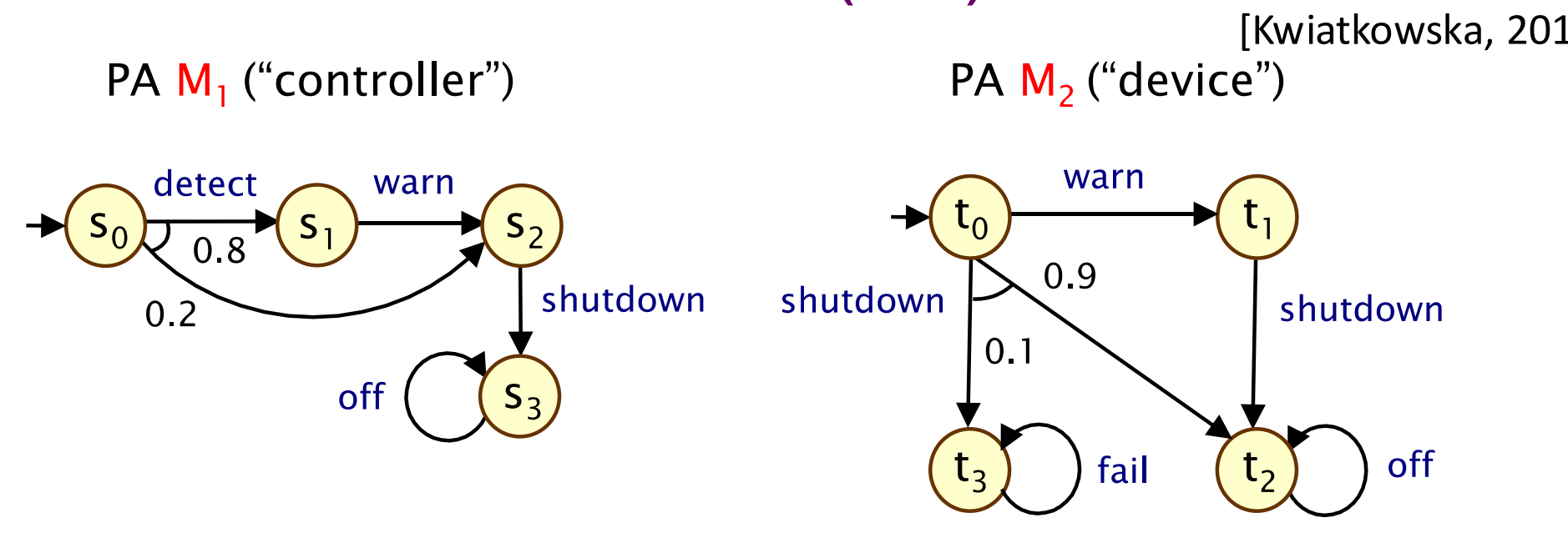
Probabilistic Verification at SIFT

Smart Information Flow Technologies

Work with collaborators from Carnegie Mellon University, University of Oxford

Probabilistic Automata (PA)

[Kwiatkowska, 2010]



$$P_{\leq 0.2}(\Diamond^{t \leq 20}\text{shutdown})$$

- Combine probabilistic and non-deterministic uncertainty.
  - Device's "shutdown vs. warn" is choice
  - t_0,shudown → $\{t_2, t_3\}$ is stochastic
- Model randomized algorithms, randomized protocols, cyber physical systems, etc.
- Verify properties in PCTL*: "there should be a less than 20% chance the system will shutdown within 20 time units.

PA Property Verification

- We focus on bounded time properties (PCTL*).
- PAs combine *stochastic* and *non-deterministic* choice.
  - E.g., a randomized message-passing algorithm, we have a stochastic element (random backoff) and a non-deterministic element (senders' choices of messages)
- As with conventional verification, we use an *adversary* to resolve non-determinism.
- We must consider the optimal adversary strategy.
  - Randomized message-passing algorithm must have good performance, with high probability, even for worst-case messages.
- Finding the optimal adversary is the same as finding the optimal policy for a Markov Decision Process (MDP).
- Standard methods rely on dynamic programming either bounded Bellman backup (finite horizon) or value or policy or value iteration (infinite horizon), or sampling.

We work in the context of Oxford's open-source PRISM probabilistic verification tool.
We have added new solution strategies and decision support to PRISM in our PRISMATIC tool.
We are now working to model human-machine systems in DARPA's Integrated Cognitive Systems (ICS) program.

PRISMATIC

Heuristic Search for Probabilistic Verification

- Strengths versus other techniques:
  - Dynamic programming: lazy enumeration of state space.
  - Sampling methods do not provide guarantees of solution quality.
  - Search guarantees to find optimal solution; allows *verification* (not just disconfirmation) of claims.
- Relies on quality of heuristic for performance: AI planning has developed high quality heuristics we can use. We use *metric disjunctive heuristic.*
- Use AND/OR optimizing search: *AO**
- Effective in MDPs with rich feature structure.

Monte Carlo Tree Search for Probabilistic Verification

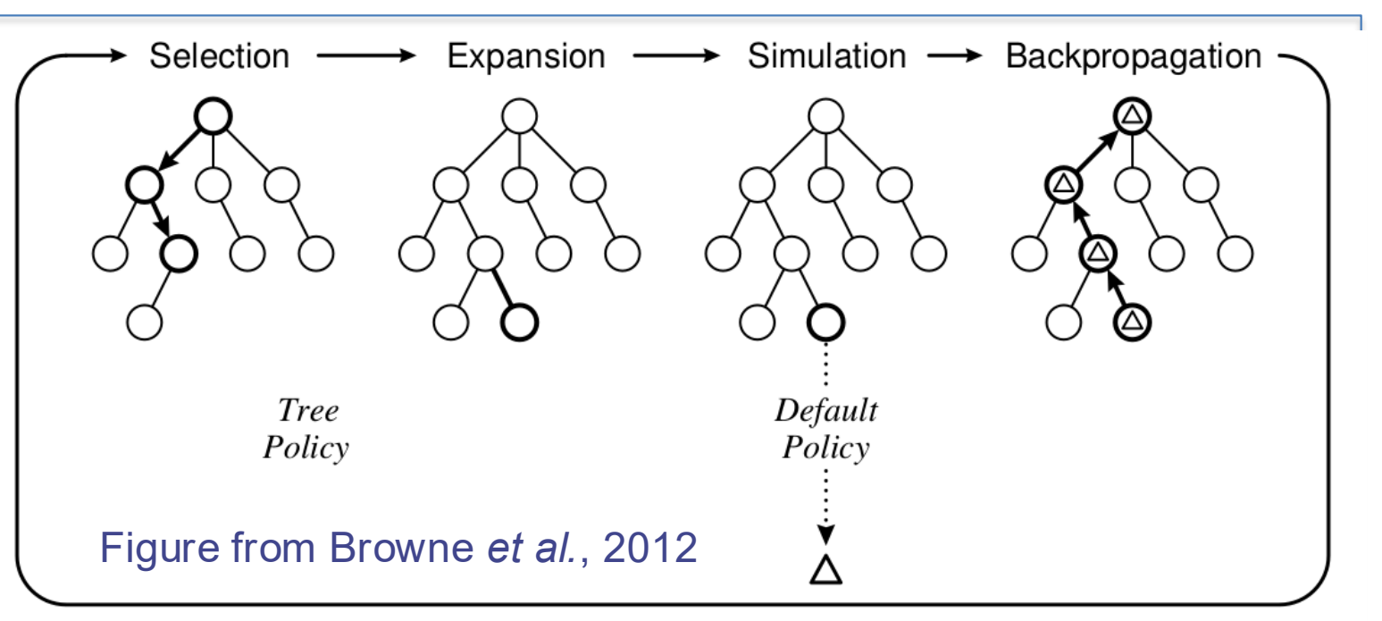


Figure from Browne *et al.*, 2012

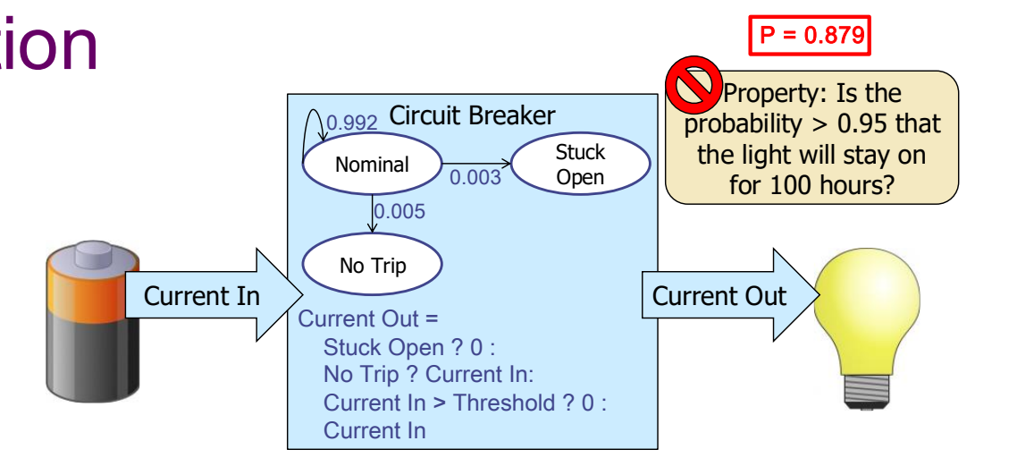
- Builds on sampling work. Issues:
  - Guarantees of convergence to best policy unproven.
  - Policies memoryless therefore non-optimal. Doesn't sustain *unreachability* claims.
- Smarter sampling.
  - MCTS balances *exploration* and *exploitation*, coverage and optimality.
  - Proven in very challenging applications.

Robert P. Goldman, Michael Boldt, and David J. Musliner

This work was supported by the Air Force Office of Scientific Research under contract FA9550- 12-1-0146 Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force or the U.S. Government

Culprit Identification/Counterexample Extraction

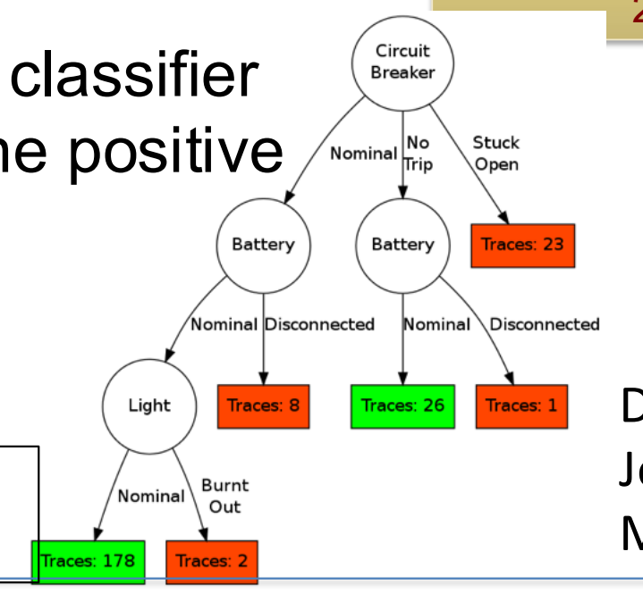
- Check system with a verifier, to see if it satisfies properties.
  - If it does: Great!
  - If it doesn't: Users want information to diagnose system failures.
- For conventional verification, get a *trace* illustrating property violation. *Relatively* simple to use.
- But for probabilistic verification
  - We have a *set of* traces that *collectively* violate the property.
  - Some violations of underlying property are OK: the overall violation is from the *probability mass* of the traces.


P = 0.879
Property: Is the probability > 0.95 that the light will stay on for 100 hours?
Circuit Breaker
0.992
Nominal
0.003
Stuck Open
0.005
No Trip
Current In
Current Out
Current Out = Stuck Open ? 0 : No Trip ? Current In: Current In > Threshold ? 0 : Current In

- Collect simulation traces through the system (positive and negative).
- Summarize each trace by a vector of component failure modes.
- From these vectors, build a classifier (decision tree) separating the positive and negative traces.

- Take traces, summarize, build decision tree, sum blame.
- Circuit Breaker 100X less likely to get stuck open improves probability of property satisfaction to 0.955.

# Traces	% Culprit	Component
23	68%	Circuit Breaker
9	26%	Battery
2	6%	Light



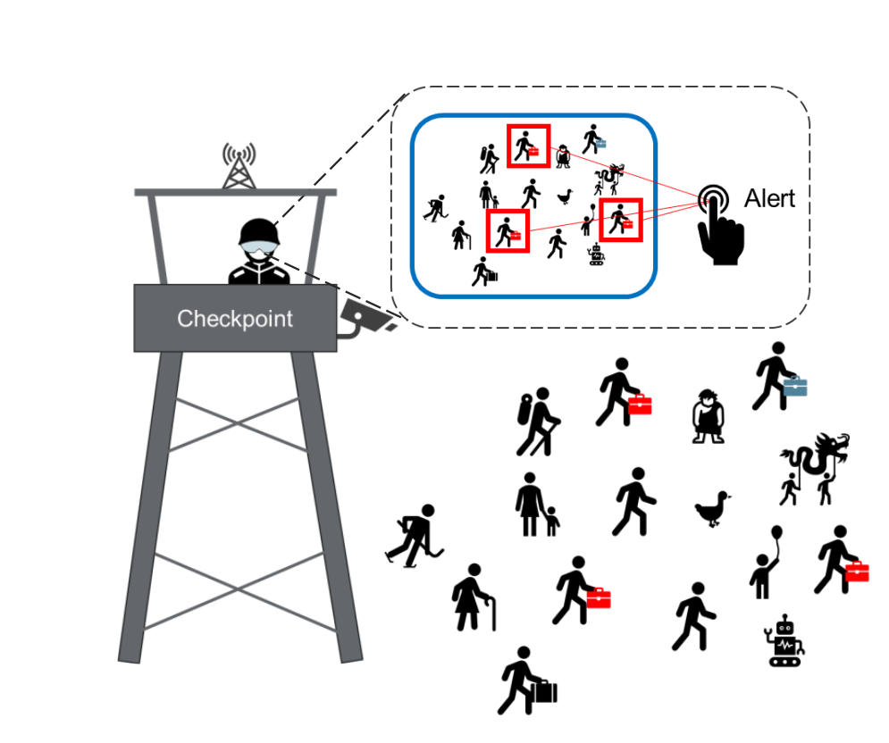
Daniel Bryce, Michael Boldt, John Maraist and David J. Musliner

This work was supported by DARPA and Air Force Research Laboratory under contract FA8650-10-C-7077. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or the U.S. Government

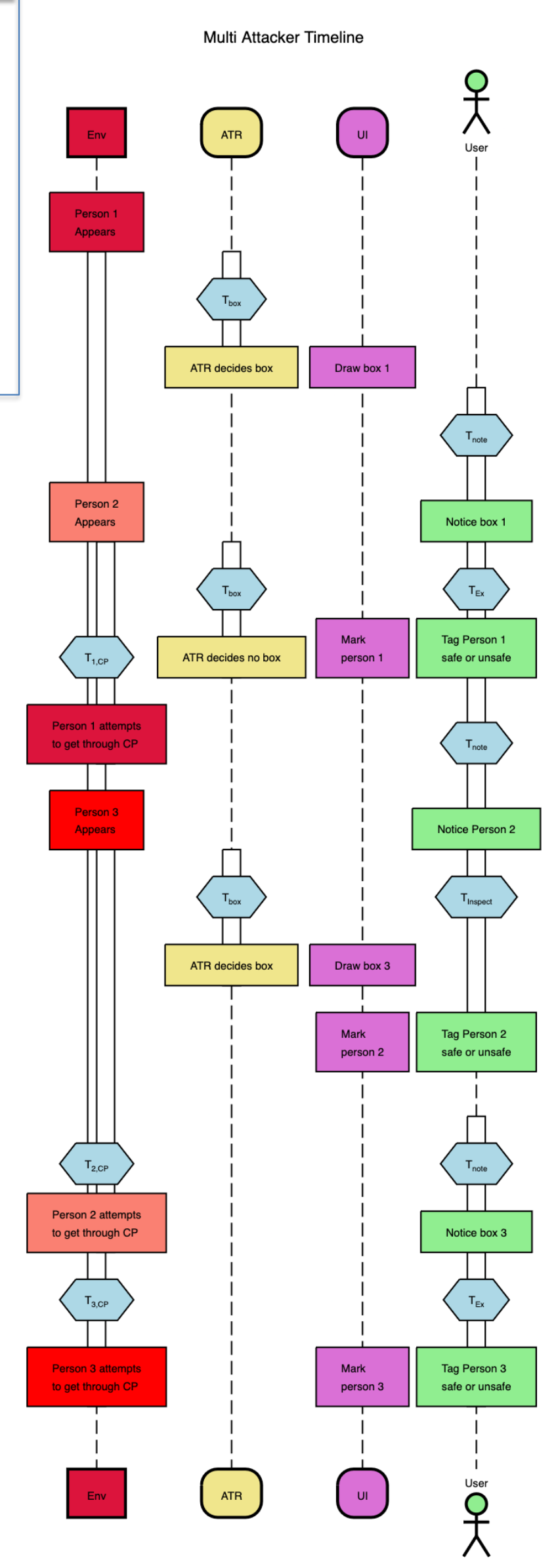
Work with RTRC, Iowa State University, Collins Aerospace An RTX Business (Prime Contractor/Team Lead), Florida Tech

This research sponsored by DARPA under contractHR0011-24-9-0439 OTA Views expressed here are solely the authors' and not those of the US government, DARPA, or the Department of Defense .

Challenge Problem:



- Private Parker uses an HMD with GuardAID to conduct surveillance
  – GuardAID highlights people with suspicious packages
  – Parker must confirm alerts and identify suspicious packages missed by the scans
- Analysis requires combining hardware and software models with psychophysics and cognitive models.
- Human in the loop leads to desire for probabilistic modeling; adversarial environment for game-theoretic non-determinism.
- Researching best method for handling temporal aspects.
- Verify that the design of the headset ensures (probabilistically) successful task performance including under attacks (e.g., presence of a large number of decoys).


Multi Attacker Timeline
Env
ATR
UI
User
Person 1 Appears
ATR decides box
Draw box 1
Notice box 1
Person 2 Appears
ATR decides no box
Mark person 1
Tag Person 1 safe or unsafe
Person 1 attempts to get through CP
Person 3 Appears
Notice Person 2
ATR decides box
Draw box 3
Mark person 2
Tag Person 2 safe or unsafe
Person 2 attempts to get through CP
Notice box 3
Person 3 attempts to get through CP
Mark person 3
Tag Person 3 safe or unsafe