From Unmanned Vehicles to Cyber Security: More Than Twenty Years of CIRCA Research Towards Trusted Autonomy

Dr. David J. Musliner
SIFT
david.musliner@musliner.com
(612) 325-9314

SIFT

The Take-Home

- Autonomy is here to stay.
- Proving safety is hard, but possible.
- Unfortunately, we want applications where safety proofs are impossible.
- This moves the challenge to defining what we really want….
  - What is “safe enough”?
  - How can we prove “safe enough”?

Presentation Outline

- Motivation.
- Architectural goals/concepts.
- Automatically synthesizing real-time controllers.
- Verifying synthesized controllers.
- Adaptive mission planning and meta-level control.
- Negotiation and planning for multiple real-time agents.
- Probabilistic reasoning.

A Tale of Two Advisors

- 1988 at the University of Michigan:

- Prof. Kang Shin
- Prof. Ed Durfee

SIFT

A Tale of Two Technologies

- Real-time Systems

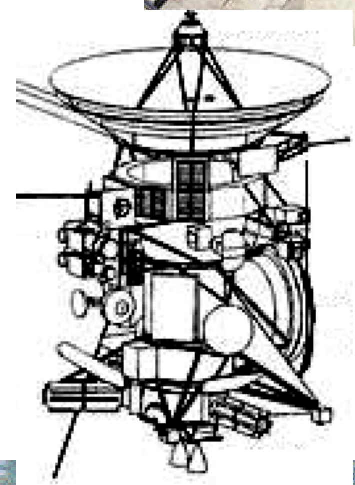
- Prof. Kang Shin

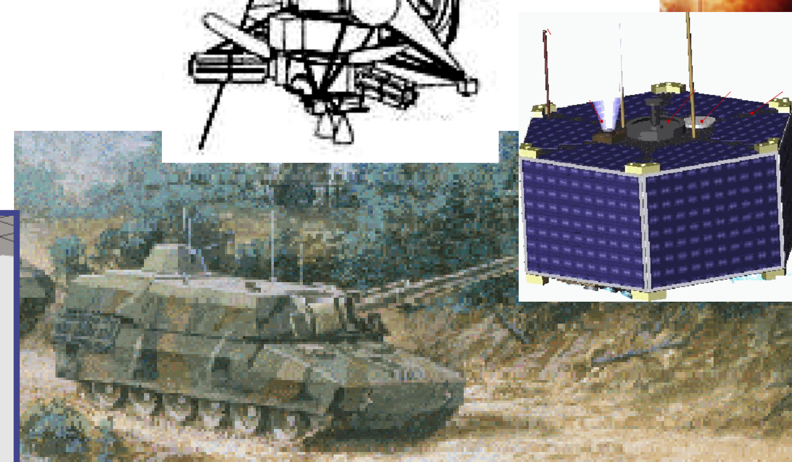
- Artificial Intelligence

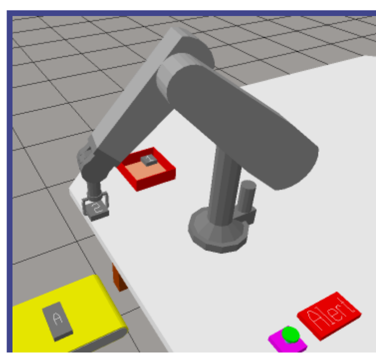
- Prof. Ed Durfee

Characteristics of Motivating Problems

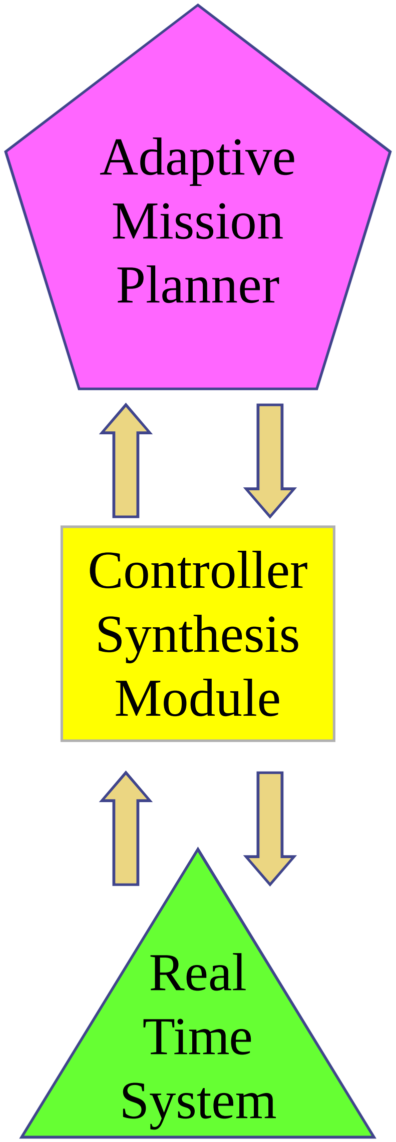
- UAVs, UGVs, UUVs, spacecraft, rovers…
- Complex dynamic environments:
  - Require intelligence.
- Critical environments:
  - Require predictable performance.
  - Logical correctness.
  - Timeliness guarantees.
- Resource limitations:
  - Bounded rationality.
  - Bounded reactivity.
- Sometimes distributed.







Cooperative Intelligent Real-time Control Architecture



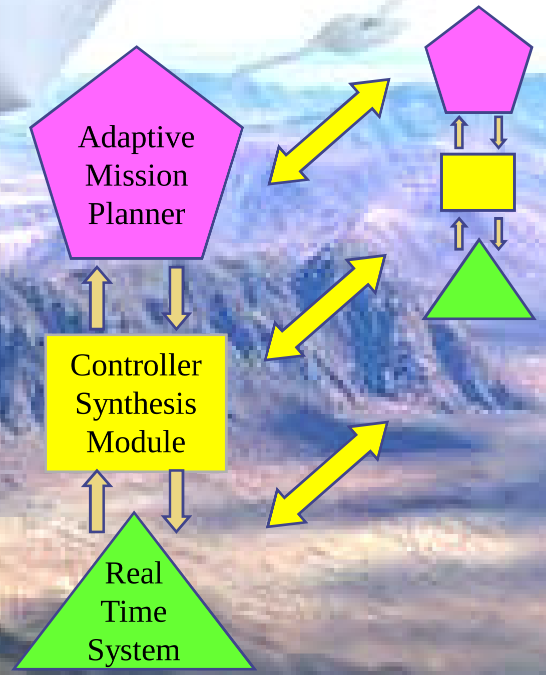
Adaptive Mission Planner: Decomposes an overall mission into multiple control problems, with limited performance goals designed to make the controller synthesis problem solvable with available time and available execution resources.

Controller Synthesis Module: For each control problem, synthesizes a real-time reactive controller according to the constraints sent from AMP.

Real Time Subsystem: Continuously executes synthesized control reactions in hard real-time environment; does not “pause” waiting for new controllers.

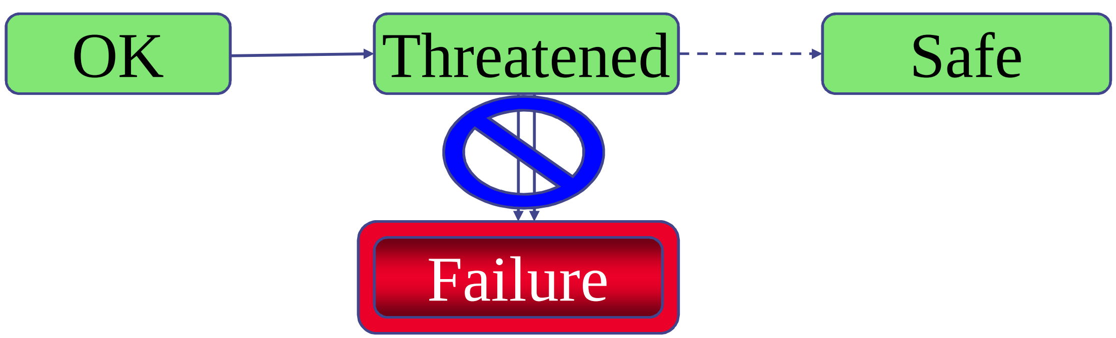
Application: CIRCA for Teams of UAVs

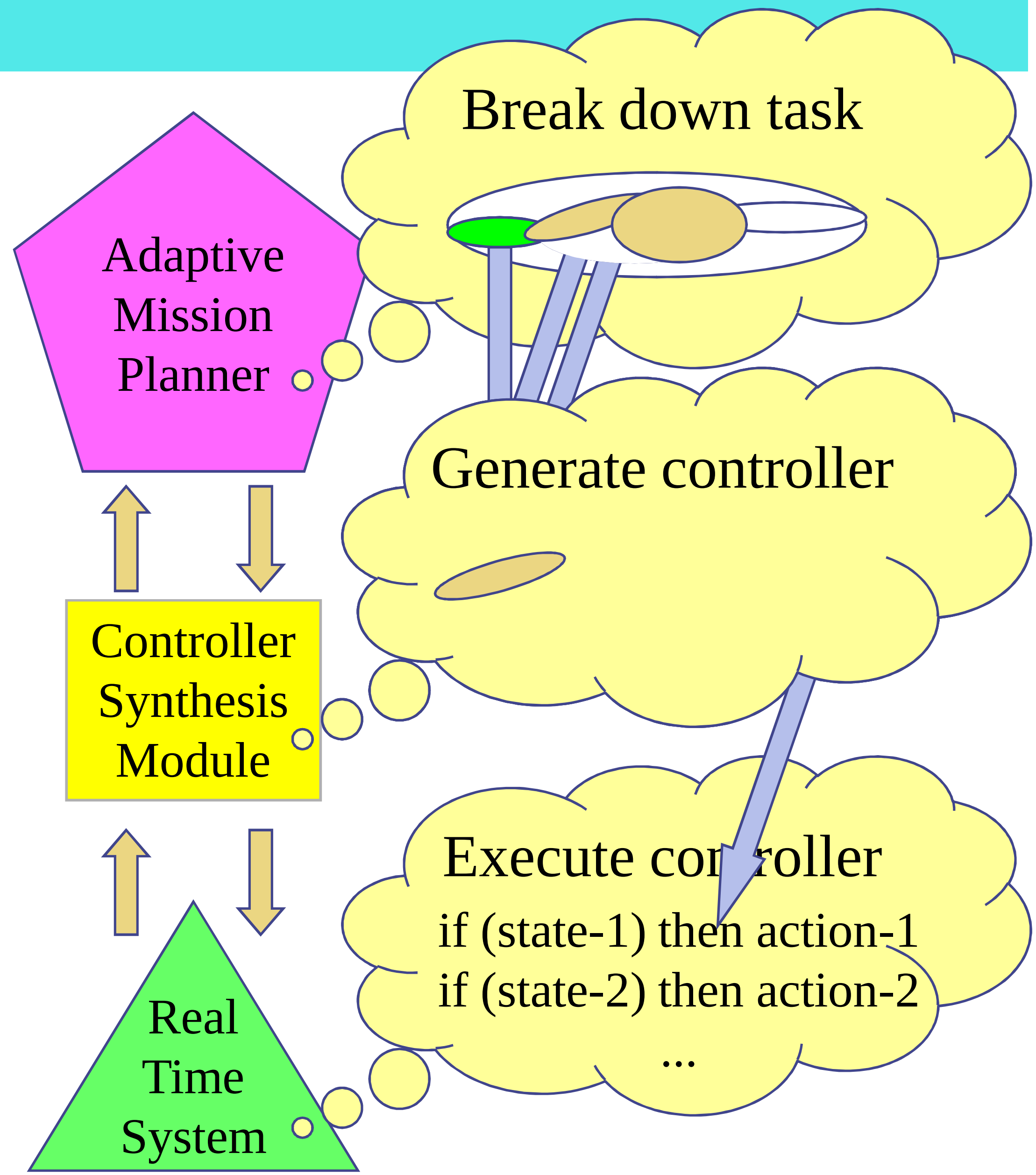
- Multi-aircraft coordinated missions/defense.
- Heterogeneous capabilities/loadout.
- Goal/system evolution.
- Real-time planning/adaptation.
- Hard RT.



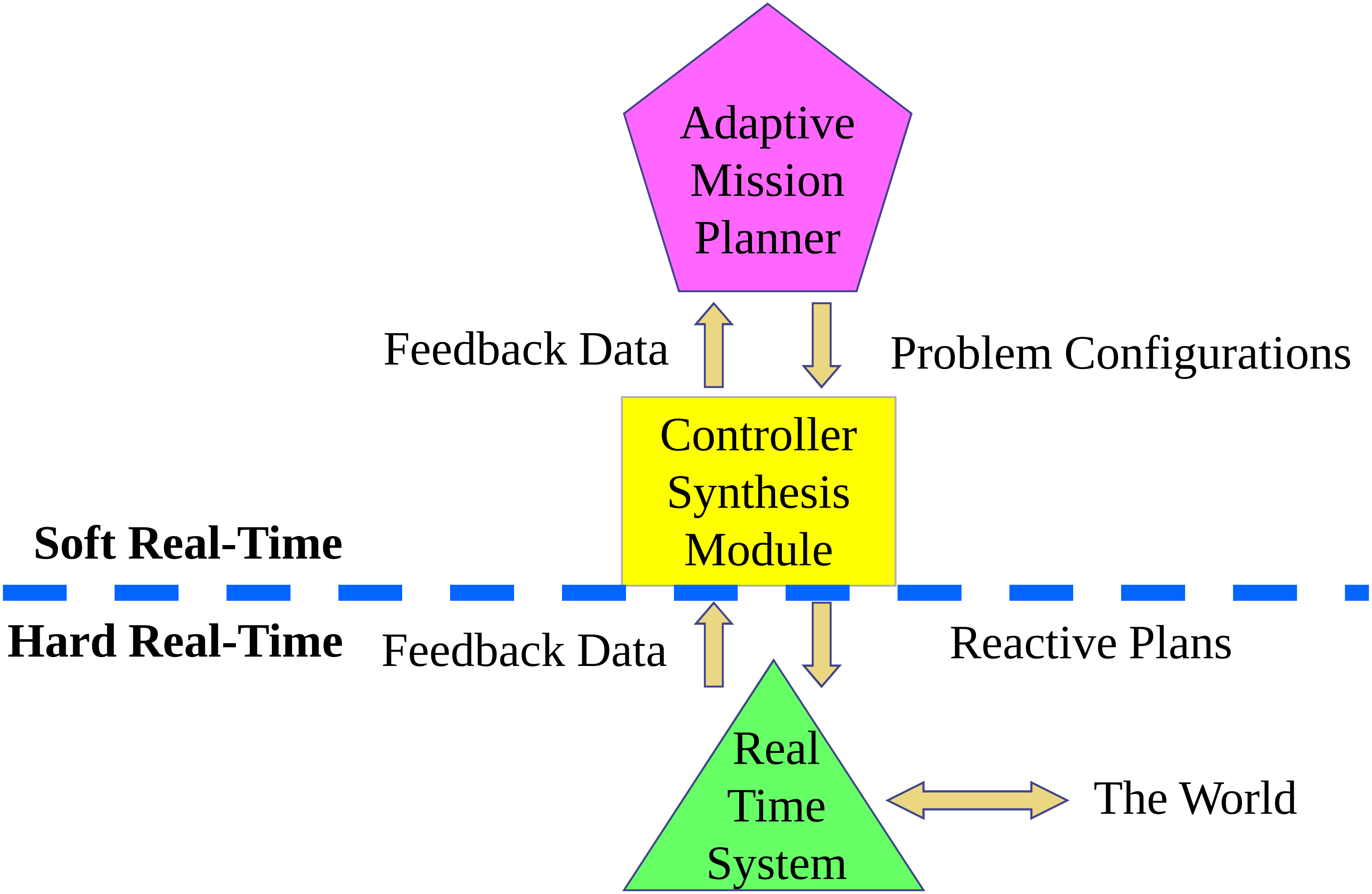
CIRCA Design Features

- Flexible systems --- CIRCA reconfigures itself *while it is operating.*
- Limited resources --- CIRCA dynamically synthesizes controllers for only the immediately relevant parts of the situation. CIRCA does this *introspectively,* reasoning about resource limits.
- Time-critical, hazardous situations --- CIRCA guarantees that it will respond in a timely way to threats in its environment.

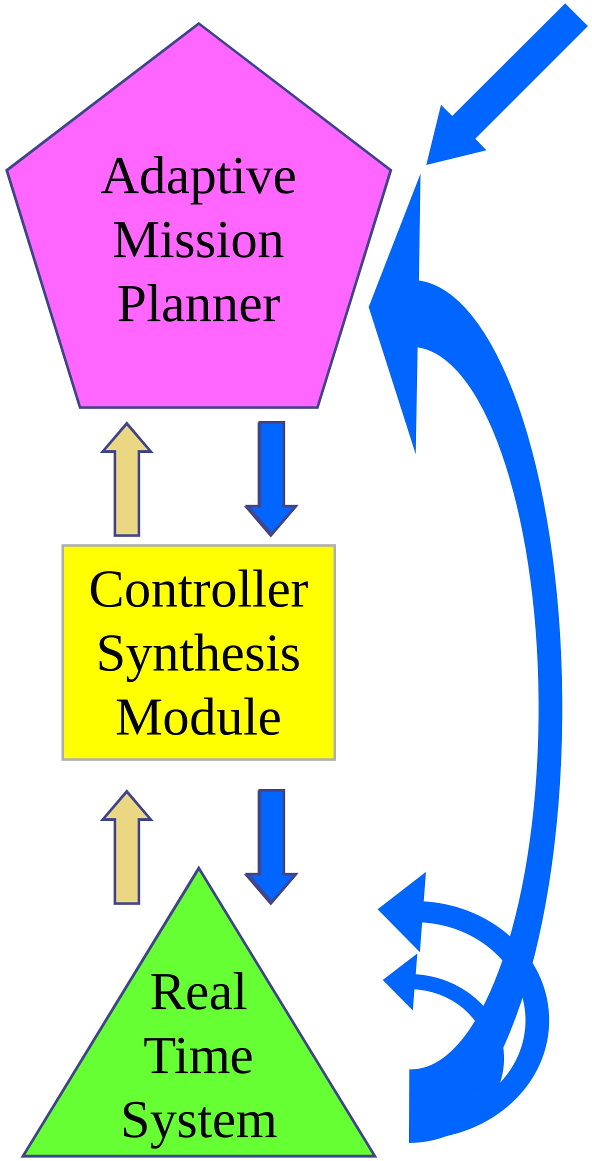



Break down task
Adaptive Mission Planner
Generate controller
Controller Synthesis Module
Execute controller
if (state-1) then action-1
if (state-2) then action-2
...
Real Time System
SIFT

CIRCA Architecture



Managing Dynamics



Time-sensitive mission planning, negotiation, and controller synthesis

Time-sensitive controller synthesis

Controller cache

Reactive control

The Main Point

Don’t program embedded real-time control systems.

Automatically synthesize them!

(and re-synthesize online to adapt)

Don’t hand-model the controller: only model the domain, goals, and system capabilities.

Online self-verification will make these systems more reliable and trustable.

Real-Time Subsystem (RTS)

Test: (AND (RADAR-GUIDED-THREAT-DETECTED T)
(HAVE-CHAFF T))

Action: (DEPLOY-CHAFF)

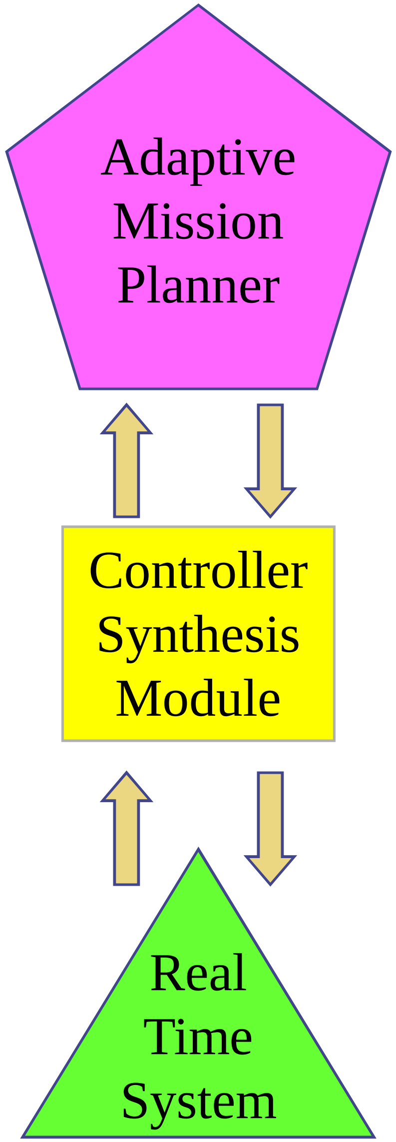
- The RTS executes a loop of Test Action Pairs (TAPs).
- Each TAP on the schedule:
  - Tests for some condition in the world, by accessing sensors or internal stored data.
  - Takes a single, atomic action if the test expression is true.
- The TAP loop is scheduled to execute different TAPs at different polling rates.
- RTS can also execute TAPs in reactive mode in response to pushed-in sensor data updates.

Real Time Subsystem (RTS)

- The RTS executes in parallel with the other CIRCA modules.
- Enforces upper bound on reaction time to anticipated situations.
- Parallel execution permits re-planning using computationally-expensive algorithms while preserving platform safety.
- Special-purpose TAPs used to download and switch to next controller.

- Low-level implementation details include:
  - Synchronous sensor data acquisition & latching to ensure coherent perceived world state.
  - Pre-allocated memory to avoid unpredictability.
  - Fully compatible with hard real-time OS requirements.

Cooperative Intelligent Real-time Control Architecture



Adaptive Mission Planner: Decomposes an overall mission into multiple control problems, with limited performance goals designed to make the controller synthesis problem solvable with available time and available execution resources.

Controller Synthesis Module: For each control problem, synthesizes a real-time reactive controller according to the constraints sent from AMP.

Real Time Subsystem: Continuously executes synthesized control reactions in hard real-time environment; does not “pause” waiting for new controllers.

SIFT

Controller Synthesis Module (CSM)

Available actions

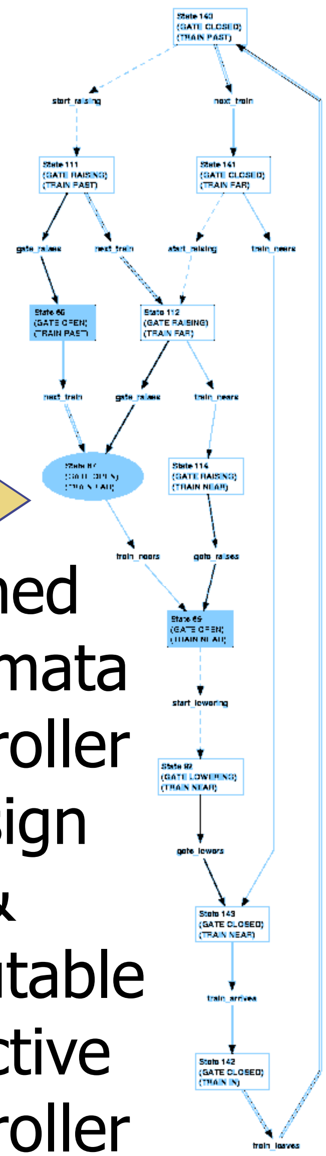
“Non-volitional” transitions

Goal state description

Initial state description

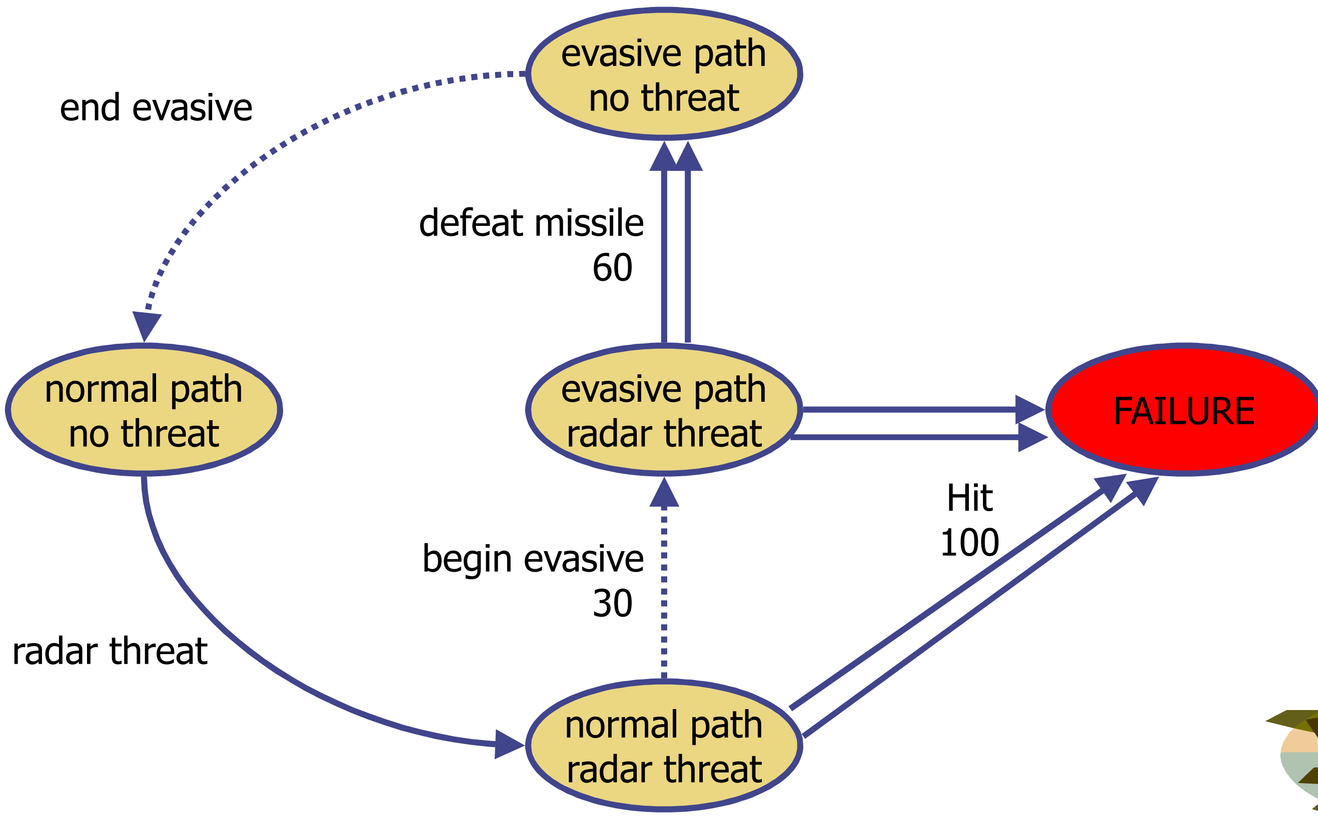
Controller Synthesis Module

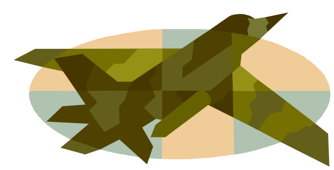
Timed Automata Controller Design & Executable Reactive Controller



Classic CIRCA World Model

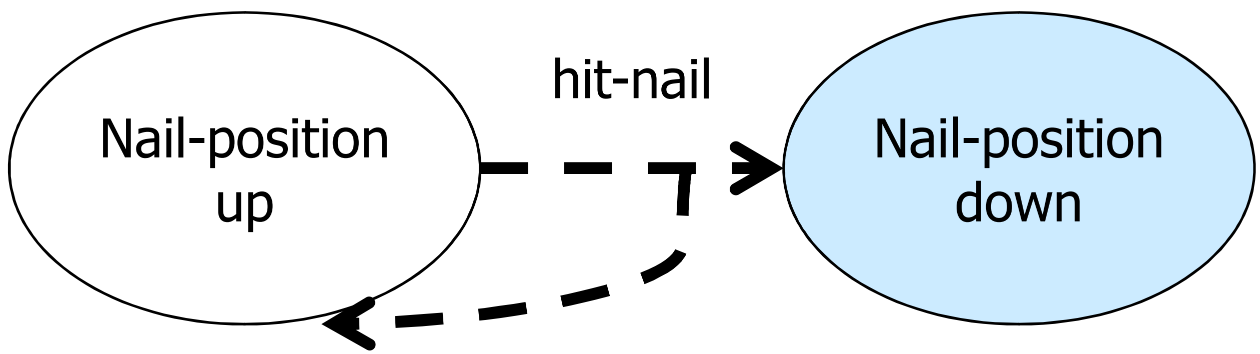
- A *plan* (or *controller*) chooses actions for states.
- Nonvolitional transitions can also move between states.
- Actions must be planned to *preempt* failure.





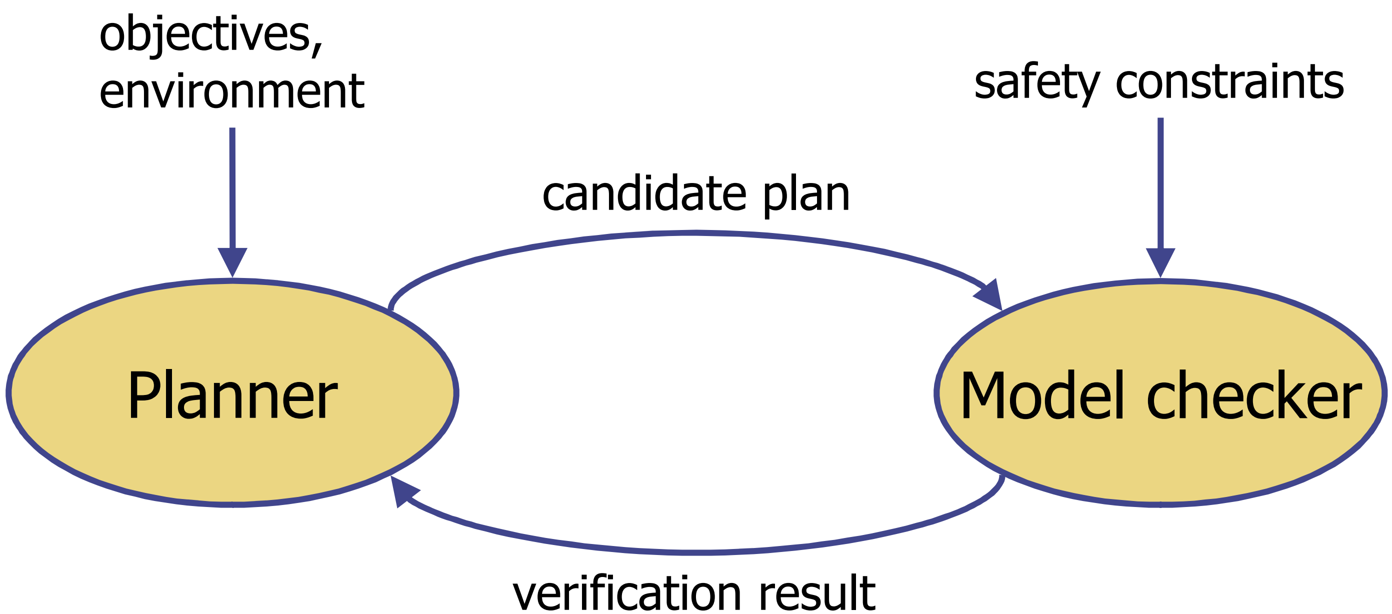
Hammering Nails

- Most AI planners build simple sequential plans as a series of reliable (perfect) actions that do not consider outside sources of change.
- CIRCA builds *controllers* that can account for unreliability and external events.



Planning with Model Checking

- State Space Planner (SSP) builds reactive controller by choosing control actions for states.
- Verifier confirms these decisions, checking that failure states are unreachable.
- If failure states are reachable, verifier provides *error trace* that directs SSP in revising controller design.
- Verification includes execution semantics of generated TAP schedule.



SIFT

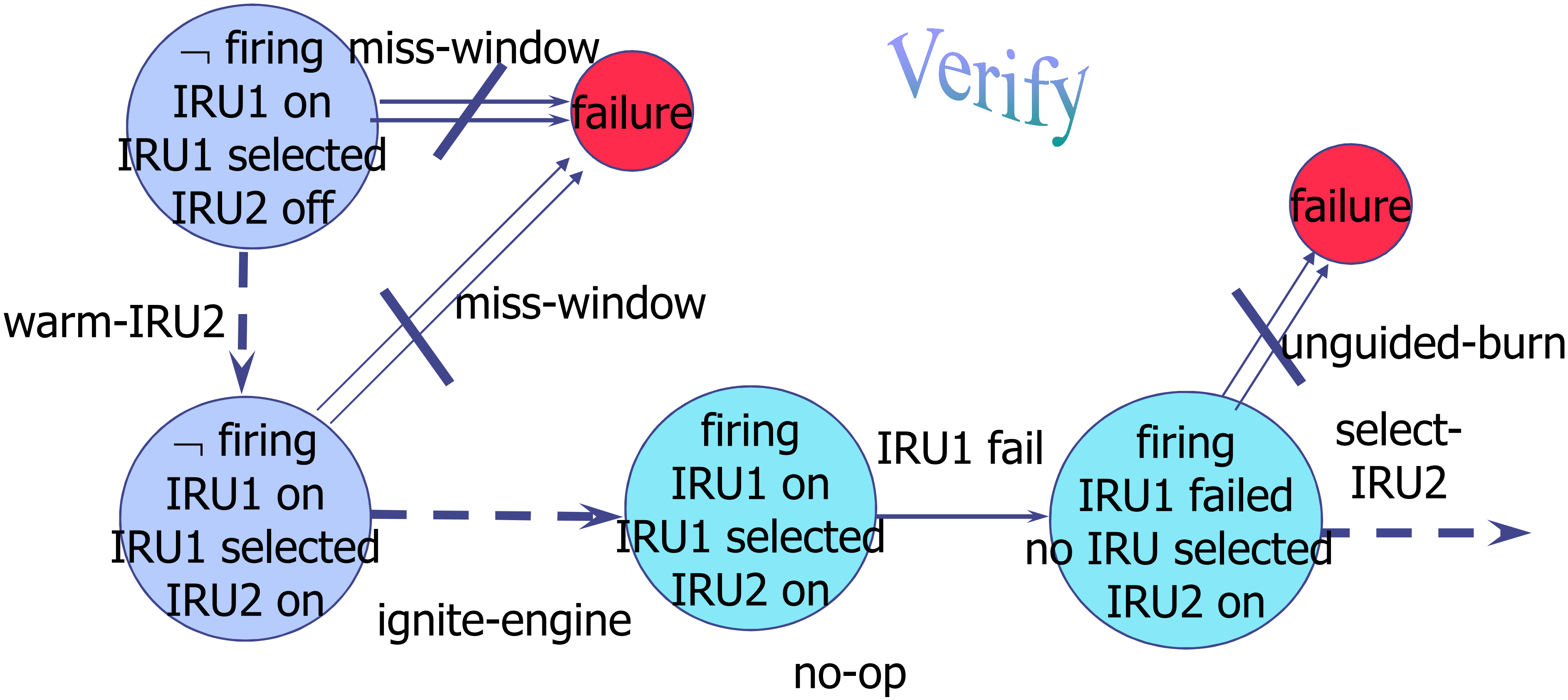
CSM Algorithm

- CSM essentially determines a strategy in a timed game against a worst-case adversary.
- Search loop iteratively selects a state and chooses action for that state.
  - Heuristics guide choice for safety and goal achievement.
  - Approximations indicate that timing will work.
  - Formal reachability analysis called after each action choice, to confirm that all planned preemptions will occur.
  - If failure reachable, path to failure can be used to backjump to most recent decision related to any state on the path.

Simplified Cassini Spacecraft Example

```
(make-instance 'action
               :name "warm_IRU1"
               :preconds '((IRU1 off))
               :postconds '((IRU1 on))
               :delay (make-range 0 1))
(make-instance 'temporal
               :name "unguided_burn"
               :preconds '((engine on)
                           (active_IRU IRU1)
                           (IRU1 broken))
               :postconds '((failure T))
               :delay (make-range 5 ∞))
(make-instance ‘event
               :name "IRU1_fails"
               :preconds '((IRU1 on))
               :postconds '((IRU1 broken)))
```

Planning with Prepositioning, Anticipating Failure



…and so forth

SIFT

Verification Background

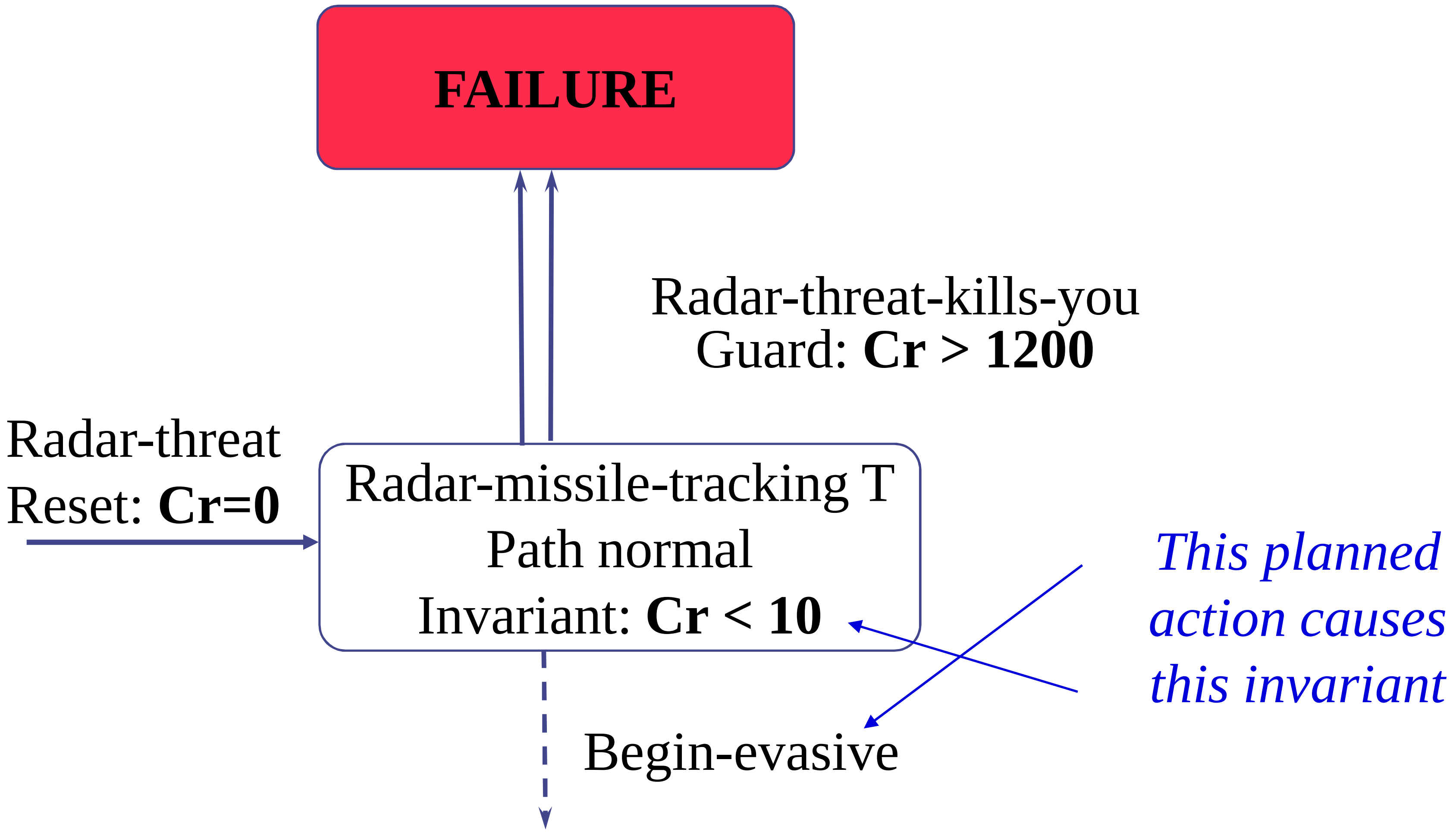
- Existing verification tools are designed for use in batch processing mode:
  - User hand-builds system model.
  - User invokes verifier, examines output.
  - User changes model based on verifier output.
  - Repeat until a satisfactory model is defined.

- CIRCA uses verification inside a fully-automatic controller synthesis cycle:
  - CSM builds partial plan and state space model.
  - CSM invokes verifier, examines output.
  - CSM changes plan and model based on verifier output.
  - CSM repeats until a satisfactory plan is defined.

Timed Automata Verifiers

- Use advanced techniques to find equivalence regions in the space of continuous clock values.
- Exhaustively enumerate the possible system traces, modulo clock region equivalence.
- Are the limiting resource for CIRCA state space planning.
- Kronos (from VERIMAG). Used in early experiments.
- Uppaal (from Uppsala/Aalborg). Not used by us to date.
- CIRCA-Specific Verifier:
  - Optimized for CIRCA problems, implicit transition-based representation of state spaces.
  - Incremental forward search for culprit, saves work when search algorithm does not backtrack.

Sample Timed Automata Fragment

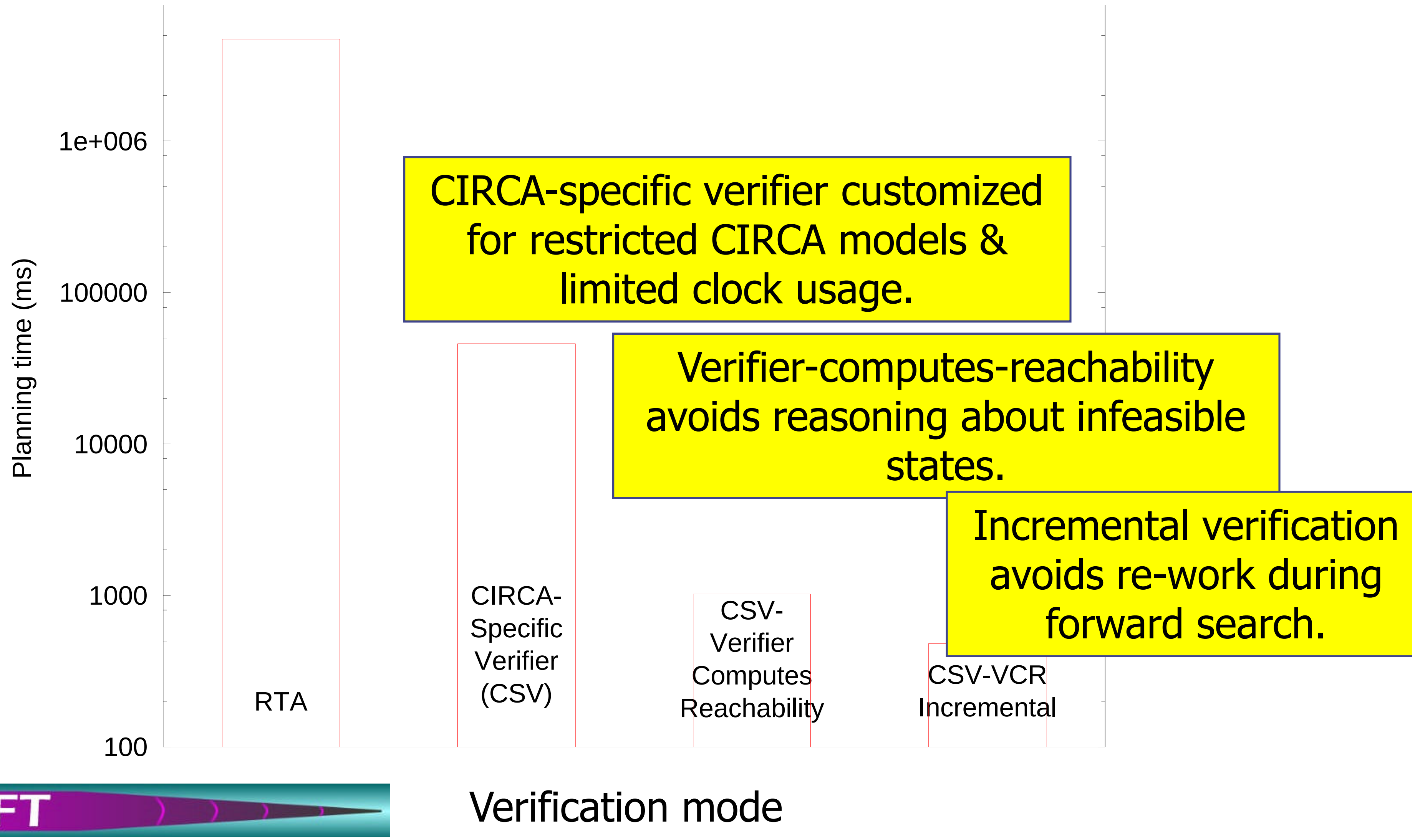


Incremental Verification

- CIRCA calls verifier many times to check ***partial*** plans.
- Problem: Model checkers can take a long time to explore all possible paths.
- Key idea: Retain information about prior verification runs to make subsequent verification runs more efficient.
  - Keep verifier traces, extend with new states as actions assigned.

Planning Algorithm Improvements

4 orders of magnitude speedup on this representative Puma domain problem



SIFT

CSM Problem Configuration Example

```
;;; Infrared Missile Threat Machine

;;; An IR-missile can start tracking you at any time.
(def-event ir-threat
        :preconds ((ir_missile_tracking F))
        :postconds ((ir_missile_tracking T)))

;;; Must respond within 1200 ticks or you will die.
(def-temporal ir-threat-kills-you
        :preconds ((ir_missile-tracking T))
        :postconds ((failure T))
        :min-delay 1200)
```

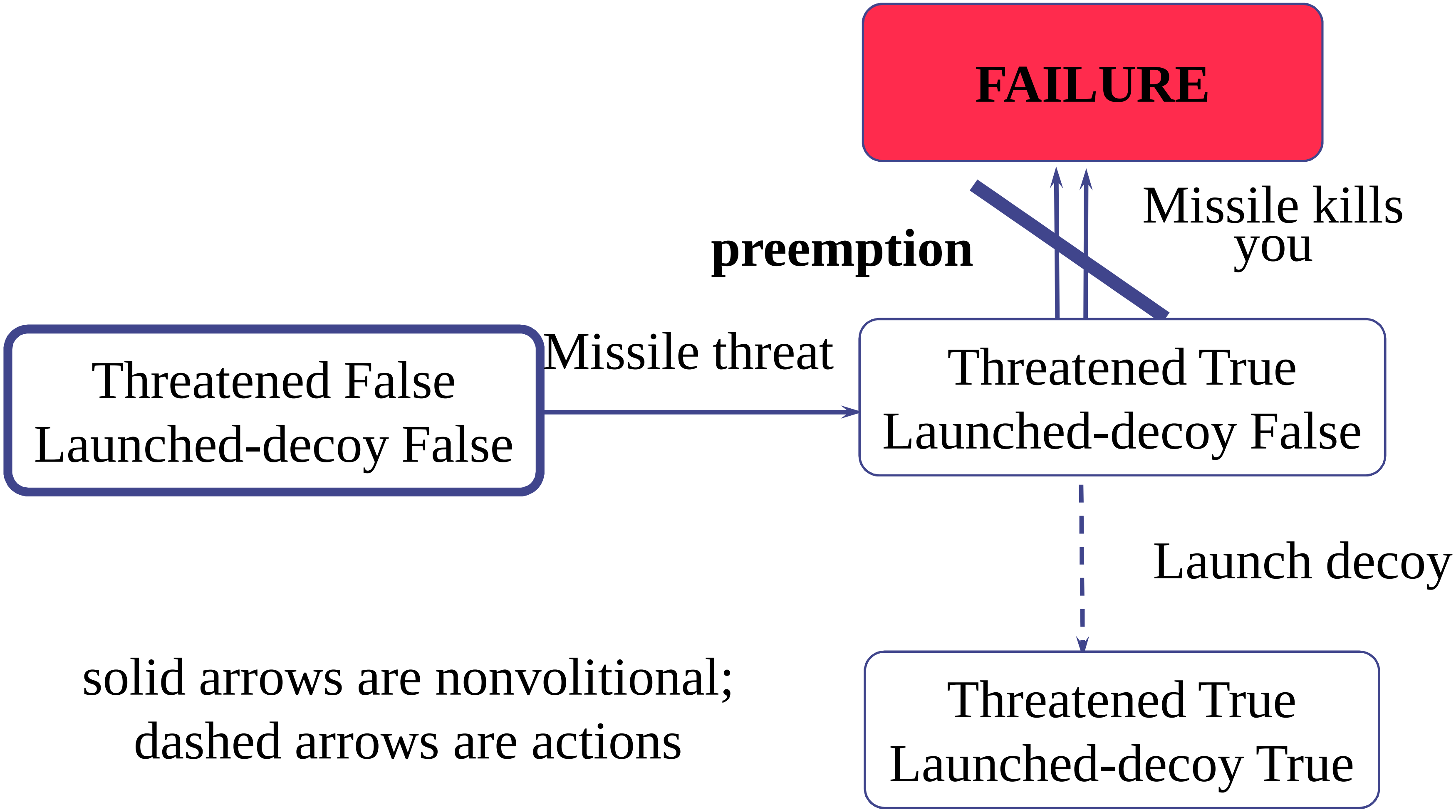
CSM Problem Configuration Example

```
;;; Infrared Missile Threat Response Machine
(def-action begin-flares
        :preconds ((flare_effective F))
        :postconds ((flare_effective T))
        :wcet 10)    ;; CIRCA will take action in 10 ticks
   or less
(def-reliable evade-ir-missile
        :preconds ((ir_missile_tracking T)
   (flare_effective T))
        :postconds ((ir_missile_tracking F))   ;; Missile
   no longer tracking
        :delay (make-range 250 400))
(def-action end-flares
```

CSM Algorithm in a Nutshell

- A search algorithm that
  - Labels each reachable state in the graph with an action that:
    - Preserves safety and
    - If possible, moves towards a goal state;
  - Re-computes the set of reachable states as actions are chosen and disturbances projected.
  - Invokes a timed-automaton verifier (e.g., Kronos) after each decision to determine whether safety is preserved (is failure state reachable?);
- Similar to timed game-theoretic approaches [Asarin, Maler, Pneuli]: choose a move for each discrete state that will avoid a victory by nature.

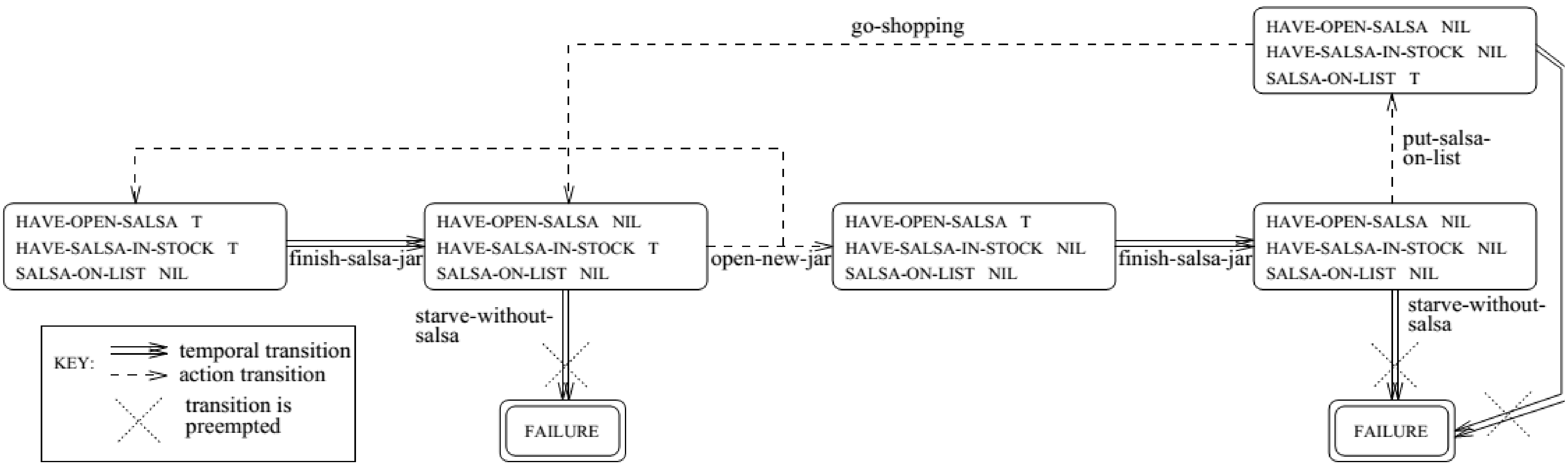
Sample Plan Fragment



solid arrows are nonvolitional;
dashed arrows are actions

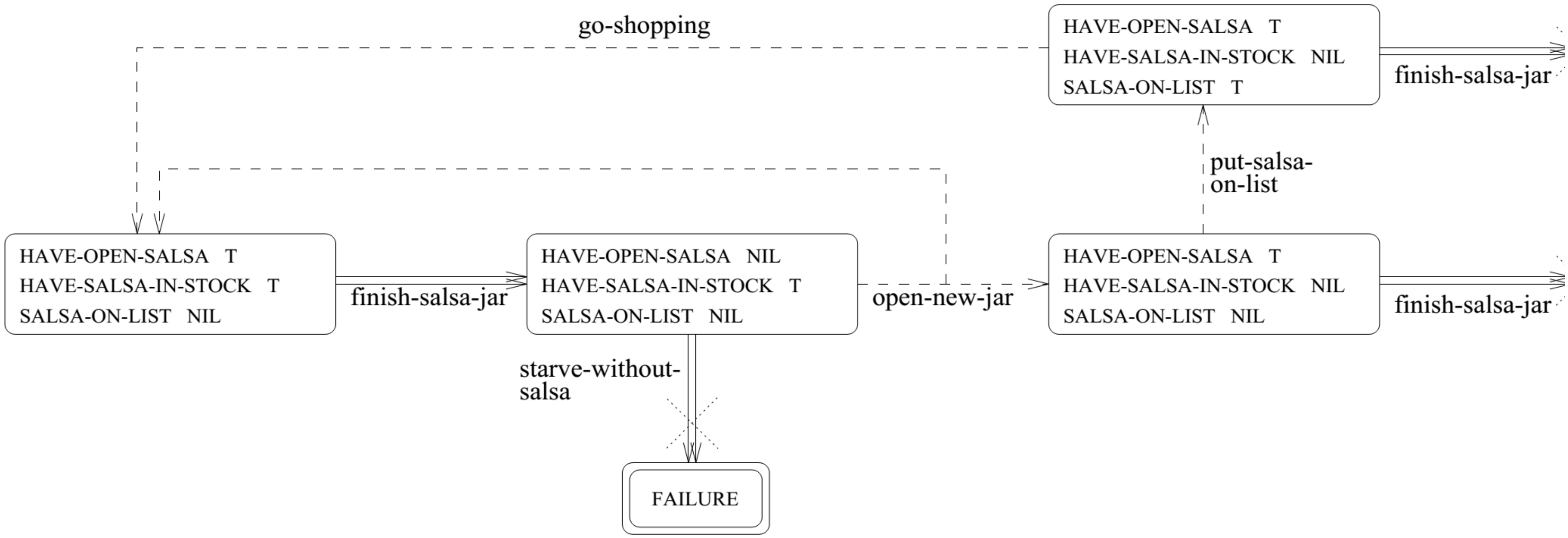
Never Run Out of Salsa

- Plan to interact with nondeterministic actions, even in the face of potential failure.
- Here: go shopping as soon as you run out of salsa.



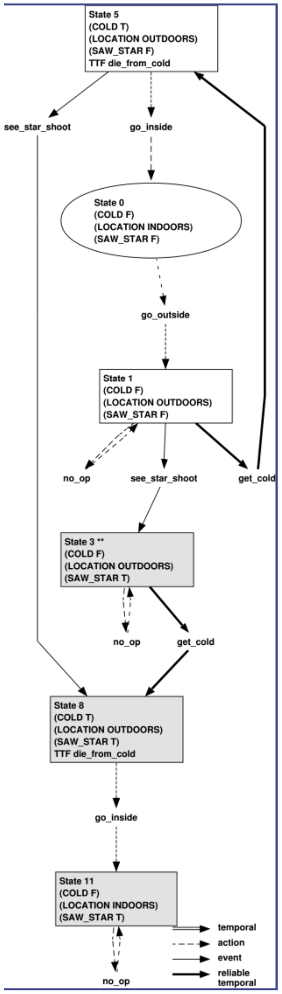
What if You Don't Shop Quickly?

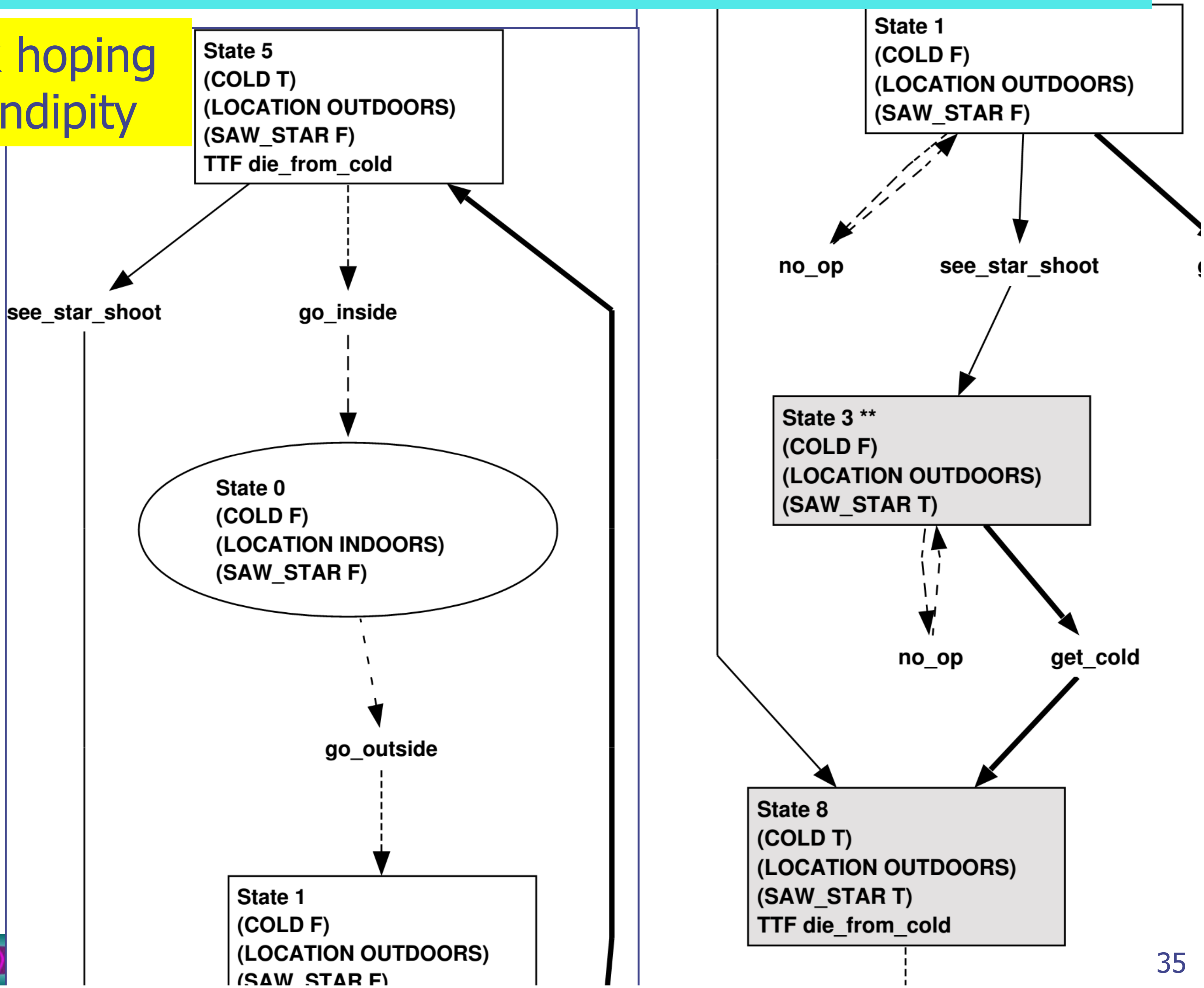
- CIRCA understands the timing: if you eat salsa too quickly, you must put it on the grocery list as soon as you open the last jar.



Seeing A Shooting Star During Minnesota Winter

Take risk hoping for serendipity





SIFT

Reminder: What's the Point?

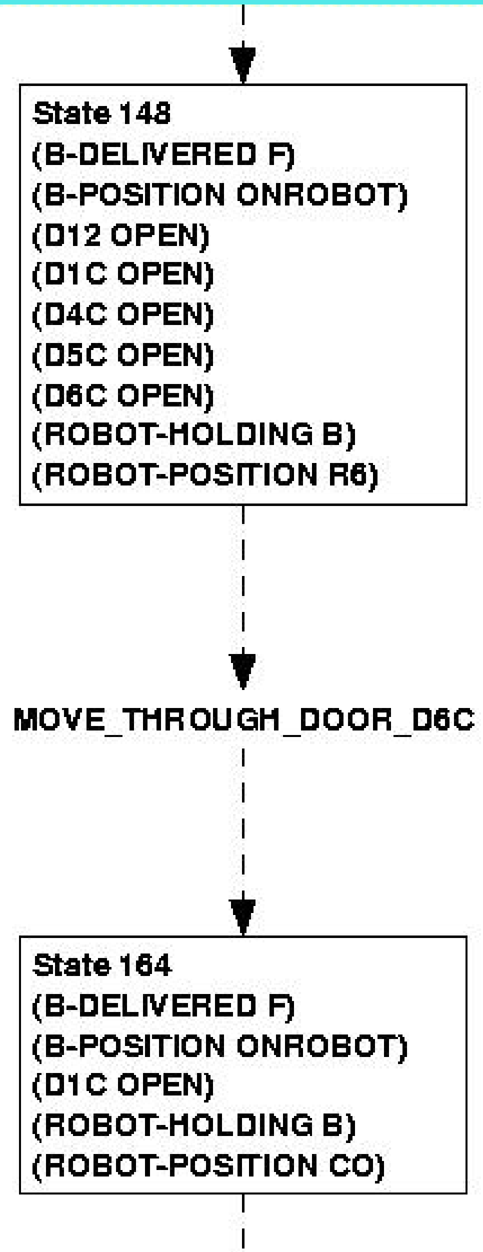
- We want to build reliable, guaranteed-safe plans that allow us to trust autonomous systems in hazardous, mission-critical environments.

- Issue: scaling… all those possible combinations of state features!

Dynamic Abstraction Planning

- Different features are important at different points in the plan.
- Represent only relevant features and represent them only when they are relevant.
- How?
- Leave out features and then locally add them back, when necessary.
  - Aka abstraction refinement.

Example DAP Plan Fragment



- Note non-homogeneous abstraction: different features specified in each state.
- Heuristic recommends:
  - Actions to take in each state.
  - “Splits” or refinements to establish action preconditions.

DAP Algorithm: Divide and Conquer

openlist **=** { {not(failure)} }

while there are reachable open nodes

choose node to plan

Either assign an action:

choose a necessarily-enabled action

consult a verifier to check the plan so far

add newly reachable states to openlist

or split state:

choose an interesting proposition and split

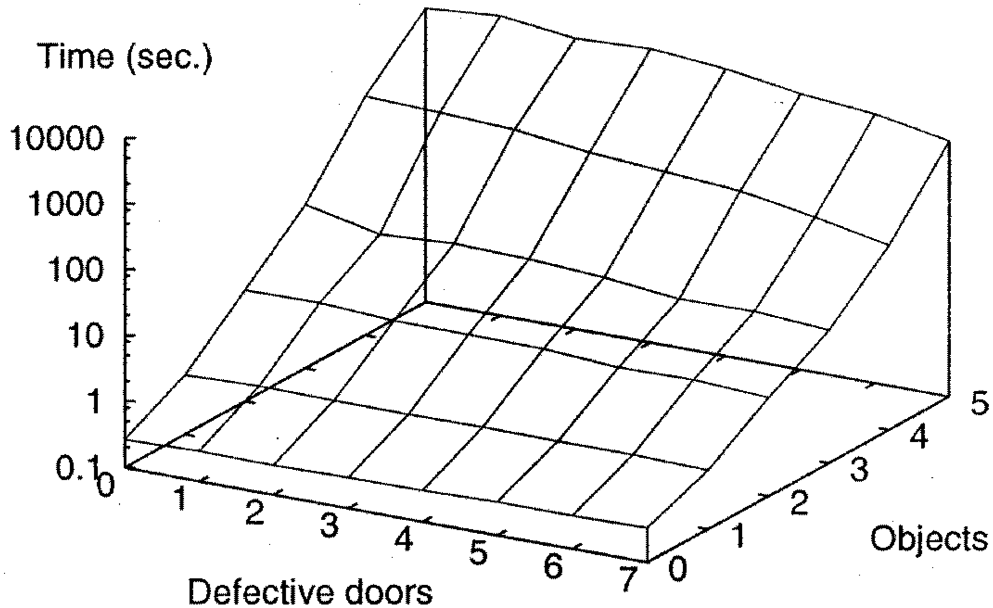
add resulting nodes to open list

endwhile

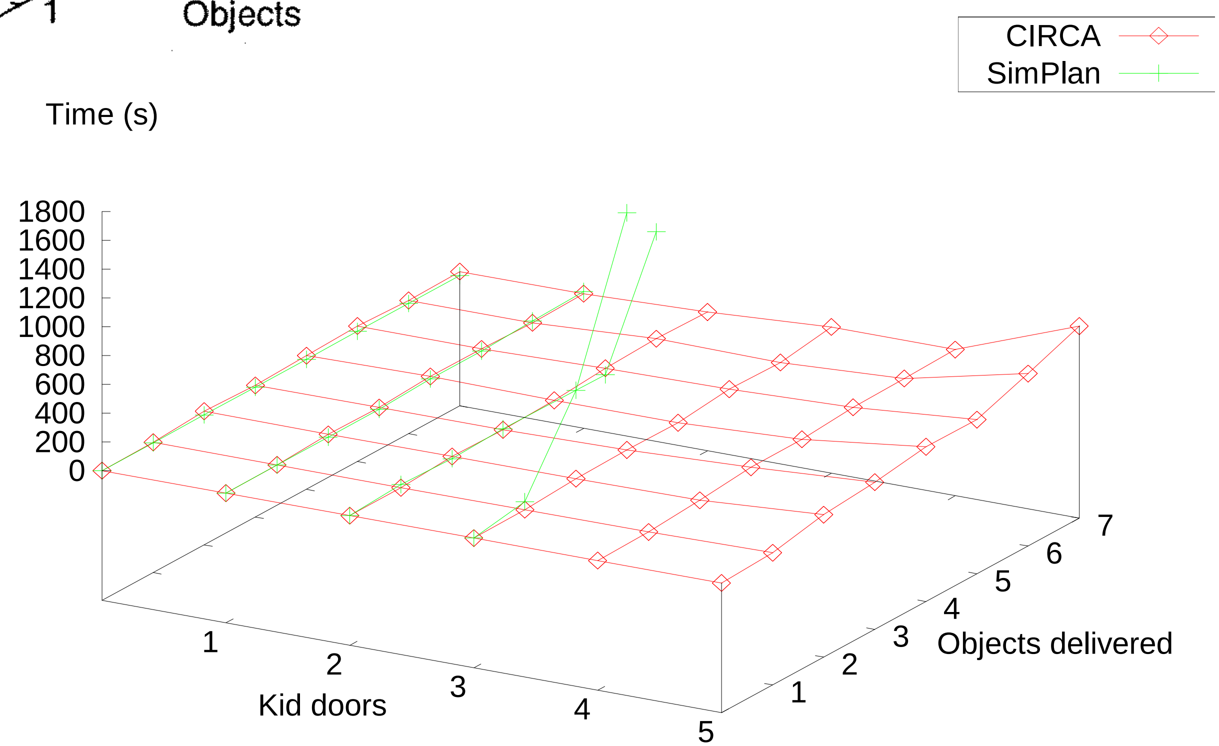
Example Domain and Related Work

- Simple robot domain used by both Kabanza and Traverso.
  - Deliver parts to rooms, including corridors.
  - Handle doors that “kid” closes (uncontrollables).
- Kabanza’s SimPlan uses *domain-specific* heuristics in forward state-space search.
  - Scales well with delivery goals, but badly with kid doors.
- Traverso’s MBP uses BDD “symbolic” state and policy representation to tackle state space explosion.
  - Scales well with kid doors, but badly with delivery goals.
- Our goal: use DAP plus ***domain independent*** heuristics to scale well on both dimensions.

Robot Domain Comparison

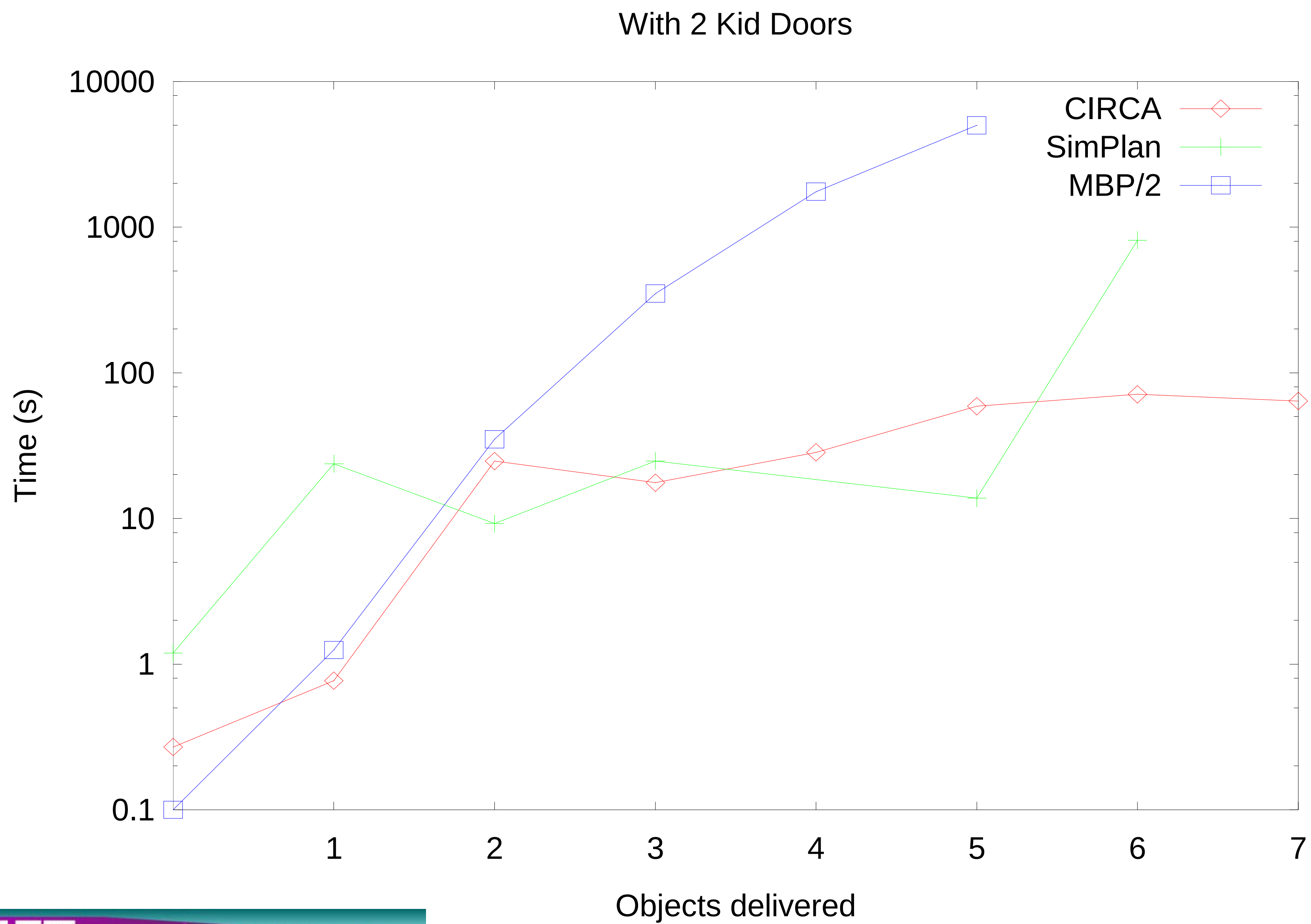


- MBP scales badly with delivery goals.
- SimPlan scales badly with both goals and doors.



SIFT

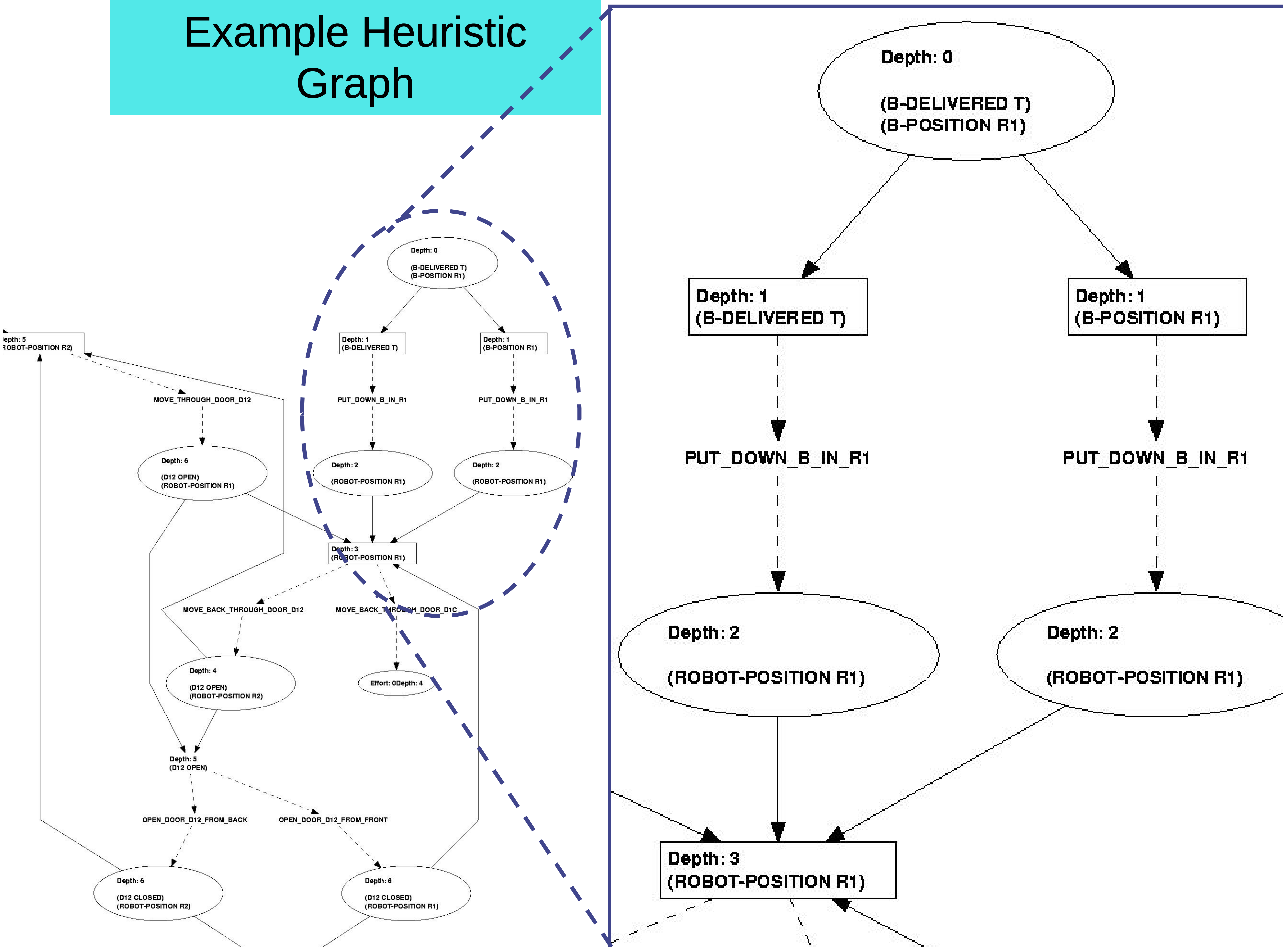
Planning Time vs Objects Delivered



Heuristic Guidance

- CIRCA's heuristic recommends:
  - Actions to take in each state.
  - "Splits" or refinements in dynamic abstraction planning (DAP).
- Based on McDermott's Unpop heuristic.
- Form a greedy regression graph linking goals backwards to current state, ignoring clobbering and sharing literal nodes.
  - Heuristic graph has cycles.
- Problem: original scoring method on graph rejects nodes that are part of cycles.
- Fails for plans with loops (e.g., robot domain with corridor).
- New: path-dependent scoring.

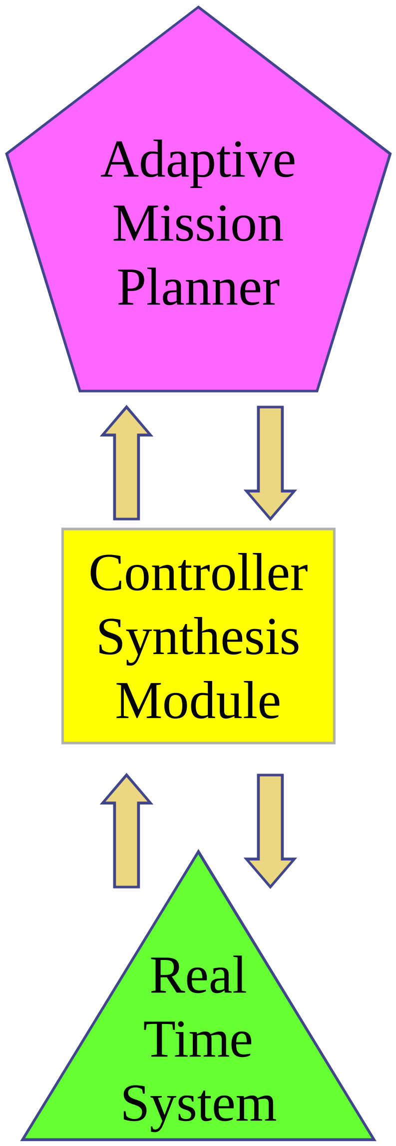
Example Heuristic Graph



Heuristic Improvements

- How choose which feature to refine (split) an abstract state on? Would like to enable some action, but which one?
- Difficulties:
  - May require more than one split to make the best action necessarily applicable.
  - Path to goal may only be apparent if splits are hypothesized to enable future actions in future states.
- Approach:
  - Hypothesize that all splits have happened, enabling all actions across all "pseudo-states" represented in heuristic graph.
  - Do this by building "fake actions" that accomplish each split.
  - Re-score graph to find best action (a real action or a fake action indicating a particular split).
- Prefer splits where:
  - The literal is in the initial state

Cooperative Intelligent Real-time Control Architecture



Adaptive Mission Planner: Decomposes an overall mission into multiple control problems, with limited performance goals designed to make the controller synthesis problem solvable with available time and available execution resources.

Controller Synthesis Module: For each control problem, synthesizes a real-time reactive controller according to the constraints sent from AMP.

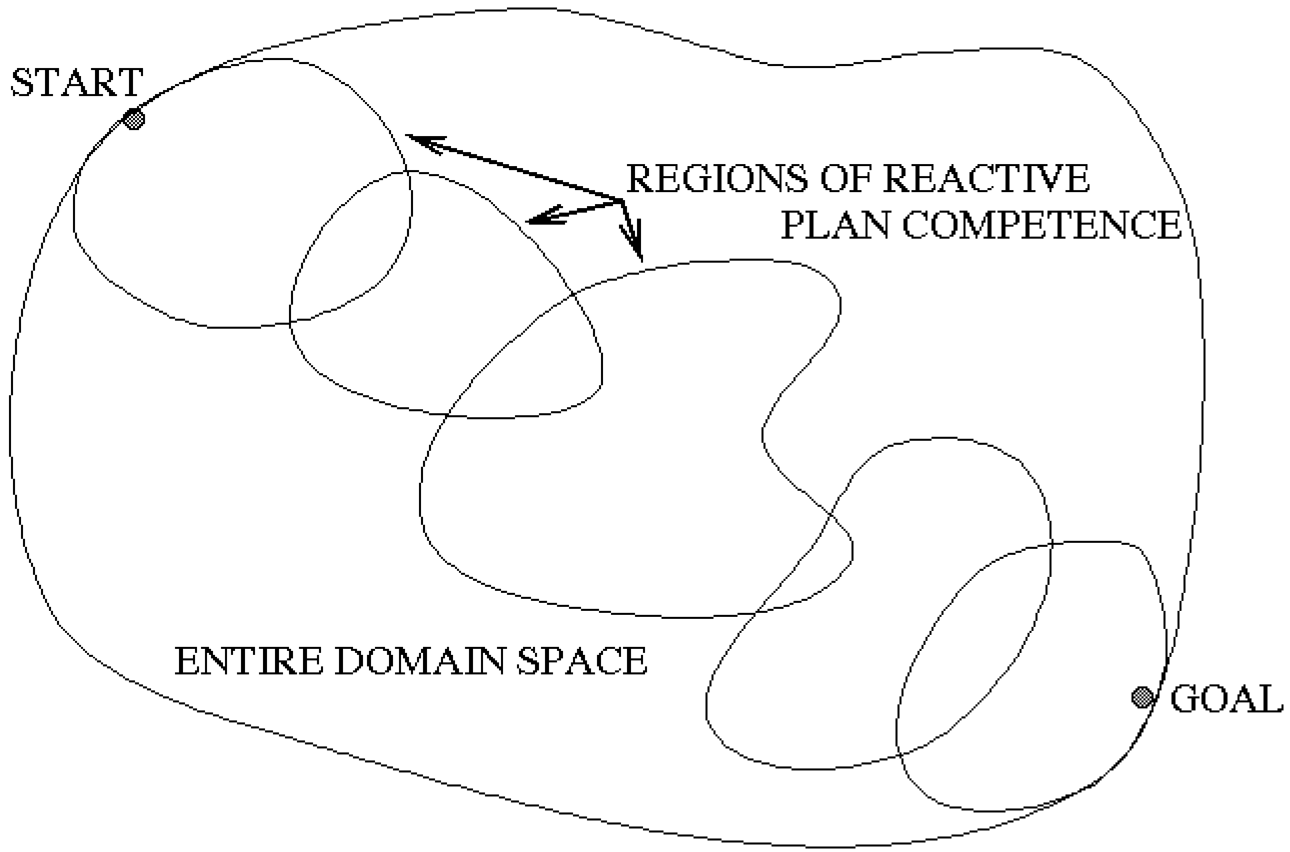
Real Time Subsystem: Continuously executes synthesized control reactions in hard real-time environment; does not "pause" waiting for new controllers.

SIFT

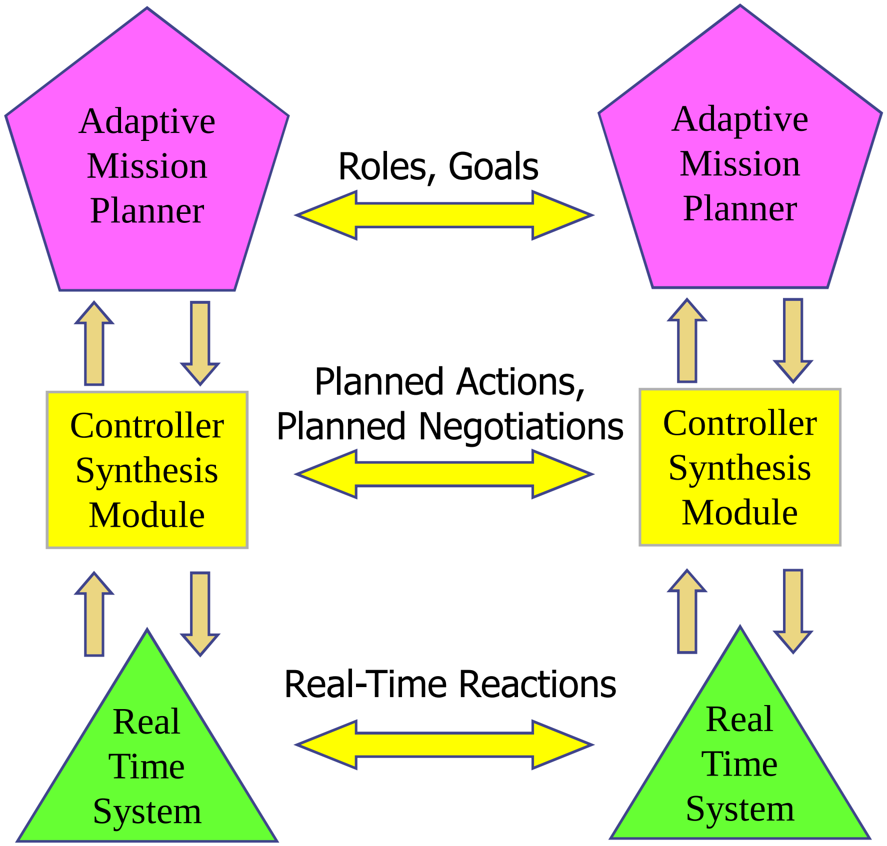
AMP Overview

- *Mission* is the main input: *threats* and *goals*, specific to different mission *phases* (e.g., ingress, attack, egress).
  - Threats are safety-critical: must guarantee to maintain safety in worst case, using real-time reactions.
  - Goals are best-effort: don't need to guarantee.
- Each mission phase requires a *plan* (or *controller*), built by the CSM to handle a *problem configuration*.
- Agents negotiate to assign responsibilities for threats/goals and build customized controllers within time bounds.
- Changes in capabilities, mission, environment can lead to need for additional negotiation and controller synthesis.

Multiple Reaction Plans



Extending Performance Guarantees to Multi-Agent Teams

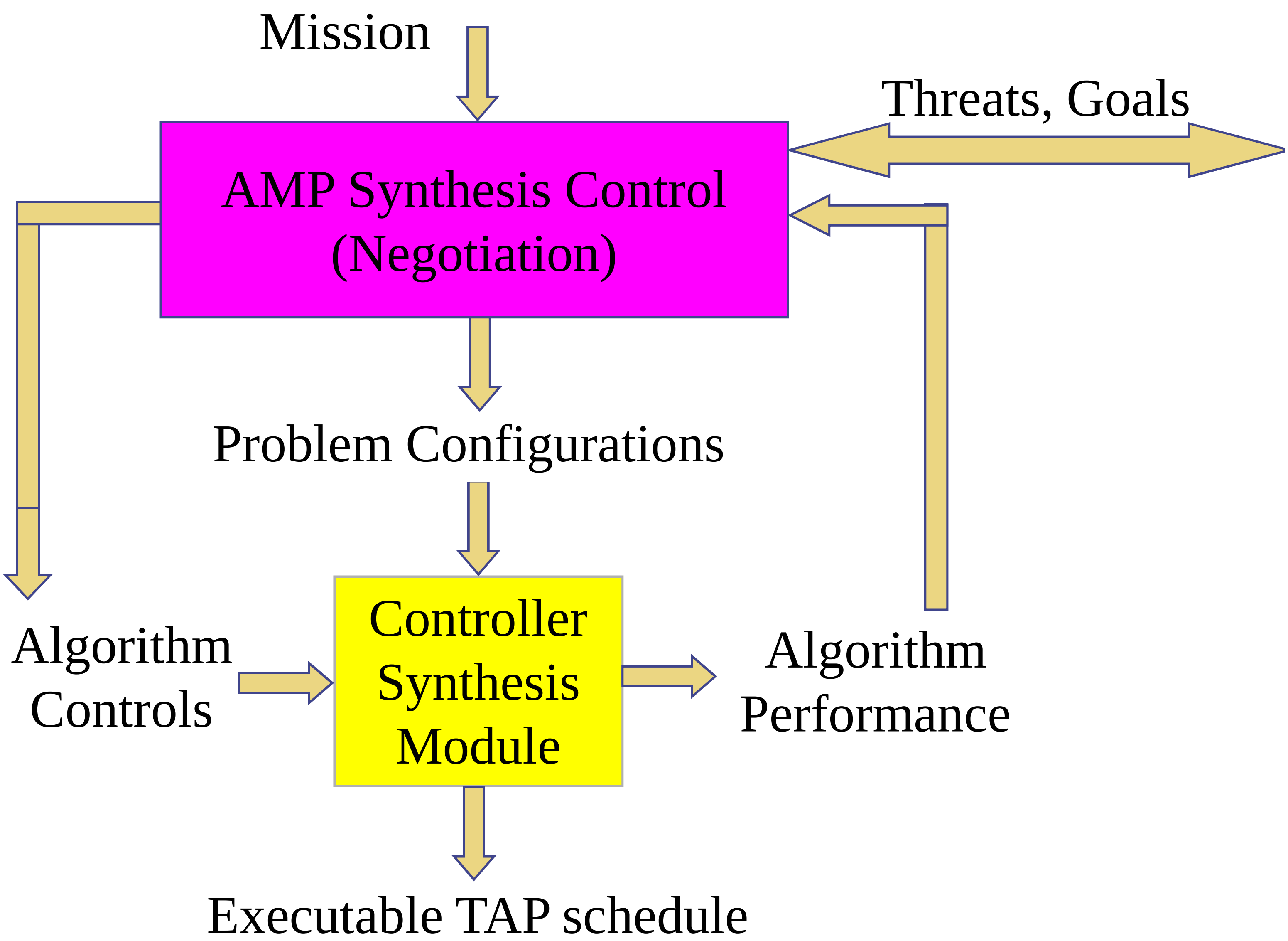


Adaptive Mission Planner:
Explicitly manages complexity of negotiation processes that dynamically distribute roles/responsibilities.

Controller Synthesis Module:
Builds controllers that include coordinated actions by multiple agents.

Real Time Subsystem:
Executes coordinated controllers predictably, including distributed sensing and acting.

AMP Overview



AMP Responsibilities

- Divide mission into phases, subdividing them as necessary to handle resource restrictions.
- Build problem configurations for each phase, to drive CSM.
- Modify problem configurations, both internally and via negotiation with other AMPs, to handle resource limitations.
  - Capabilities (assets).
  - Bounded rationality: deliberation resources.
  - Bounded reactivity: execution resources.
- Enhanced Contract-Net style negotiation to distribute:
  - Long-term mission goals.
  - Roles: predefine responsibilities/concerns as context for negotiation.
  - Performance evaluation responsibilities.

AMP Deliberation Scheduling

- Seeking principled, practical method for AMP to adjust CSM problem configurations and algorithm parameters to maximize expected utility of deliberation.
- Issues:
  - Complex utility function for overall mission plan.
  - Survival dependencies between sequenced controllers.
  - Lack of CSM algorithm performance profiles.
  - Potentially large search space of possible AMP tradeoff approaches.
  - Planning that is expected to complete further in the future must be discounted.

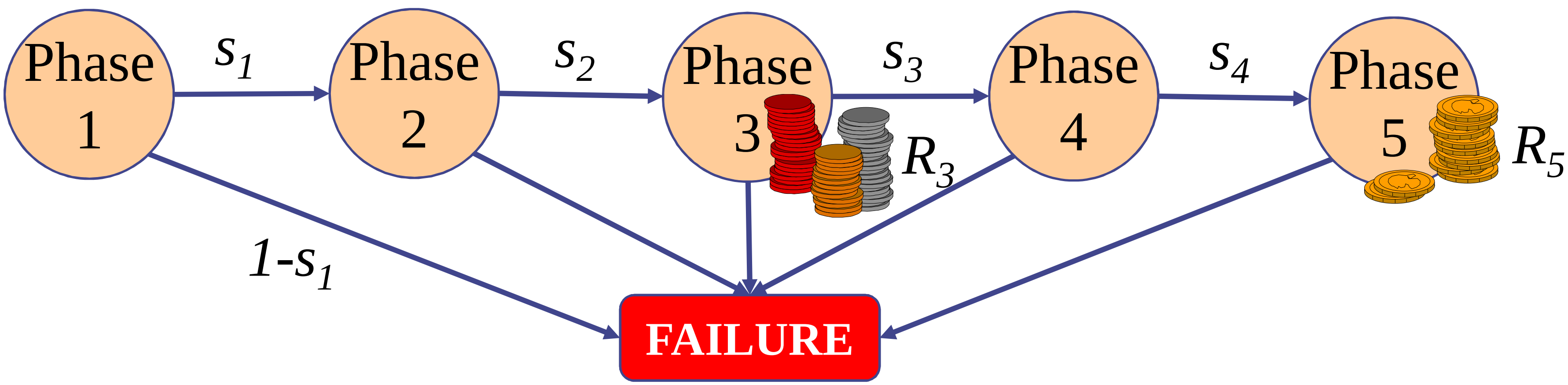
AMP Deliberation Scheduling

- Mission phases characterized by:
  - Probability of survival/failure.
  - Expected reward.
  - Expected start time and duration.
- Agent keeps reward from all executed phases.
- Different CSM problem configuration operators yield different types of plan improvements.
  - Improve probability of survival.
  - Improve expected reward (number or likelihood of goals).
- Configuration operators can be applied to same phase in different ways (via parameters).
- Configuration operators have different expected resource requirements (computation time/space).

Expected Mission Utility

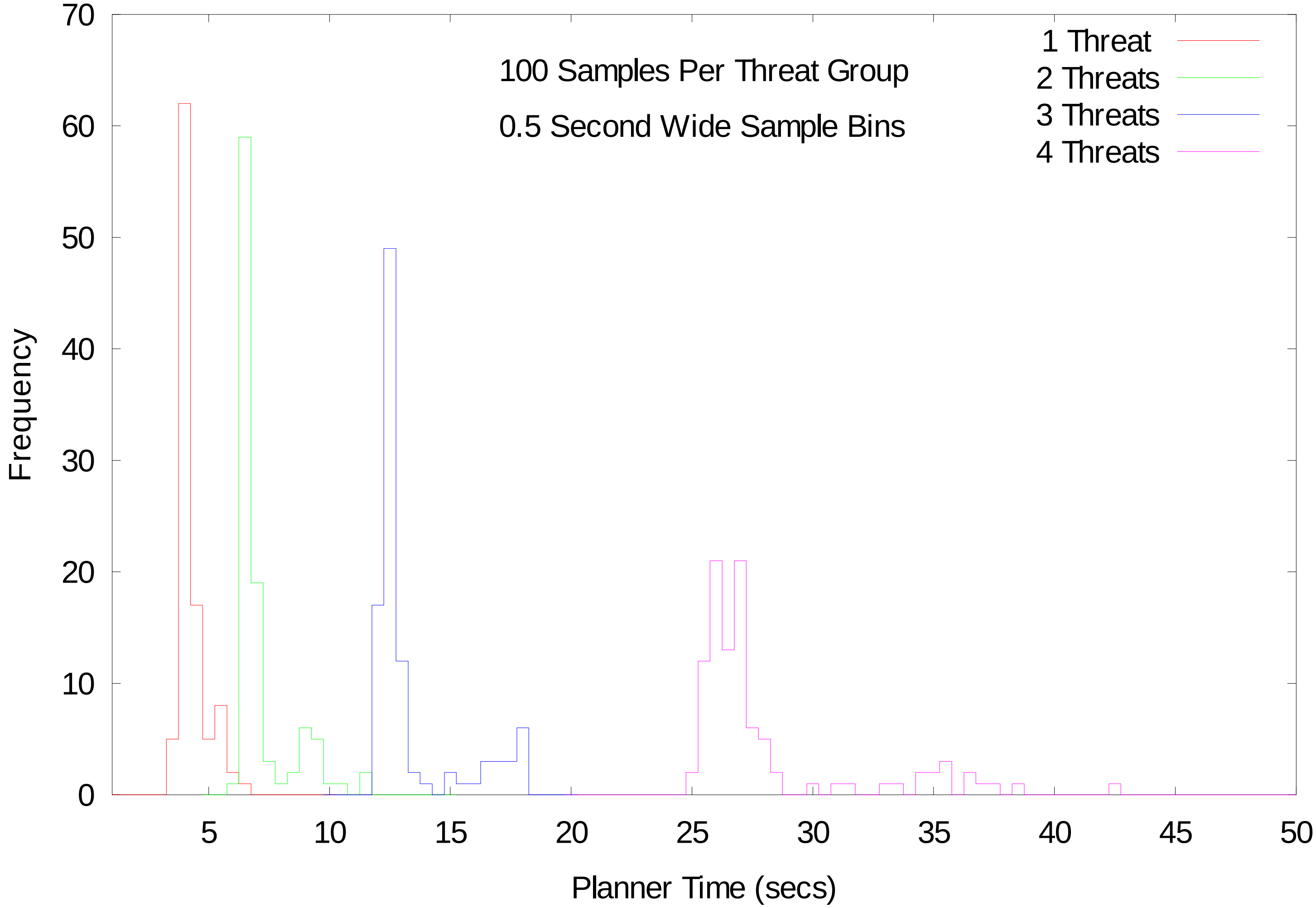
Markov chain behavior in the mission phases:

- Probability of surviving vs. entering absorbing failure state.
- Reward expectations unevenly distributed.



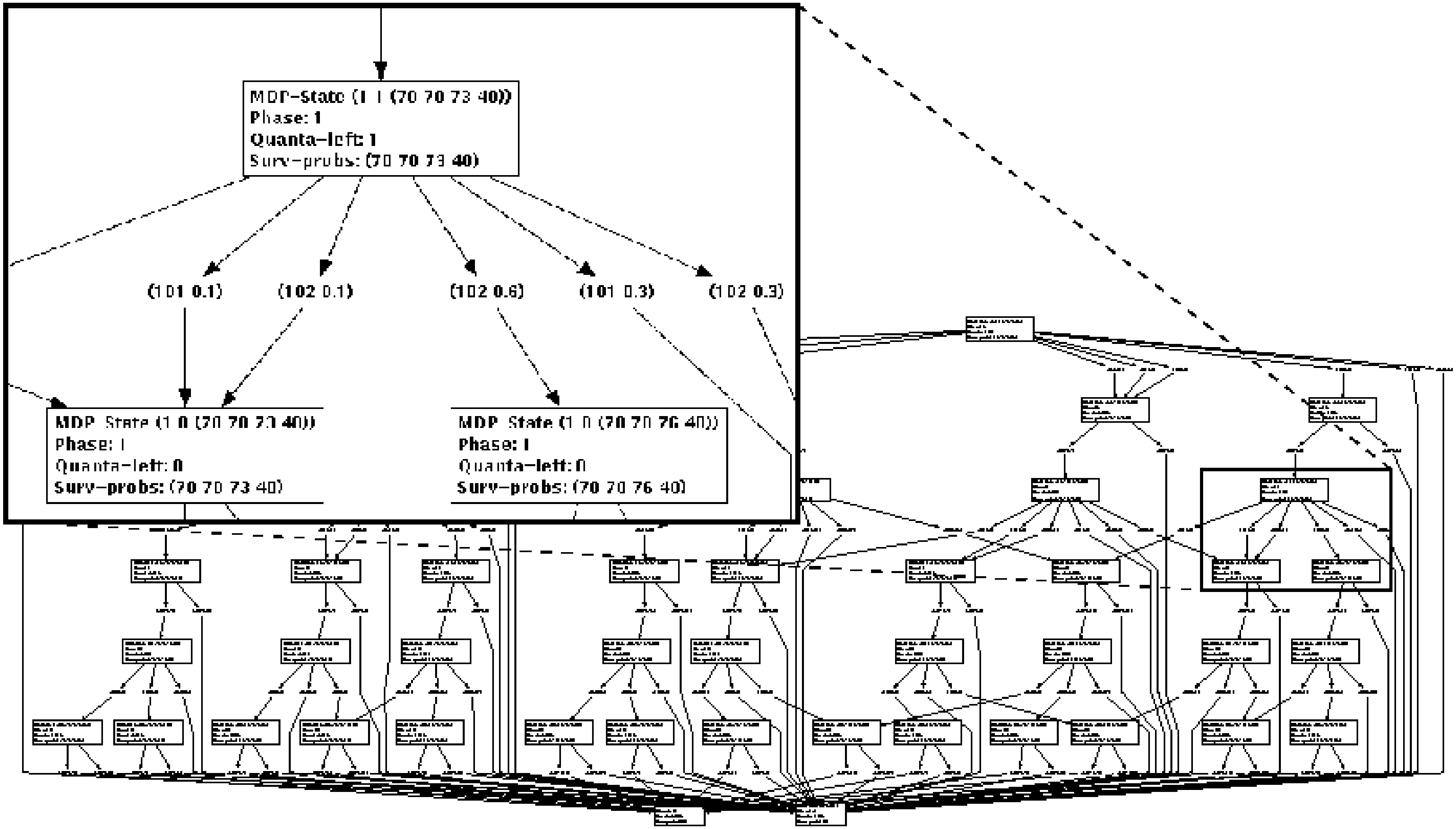
$$U = \sum_{i=1}^{n} R_i s_j$$

Histogram of CSM Performance Results



CSM runtime is moderately predictable based on number of threats
Note increasing spread (uncertainty of runtime) as problem grows.

Example Deliberation Scheduling MDP Model



Optimal Deliberation Management

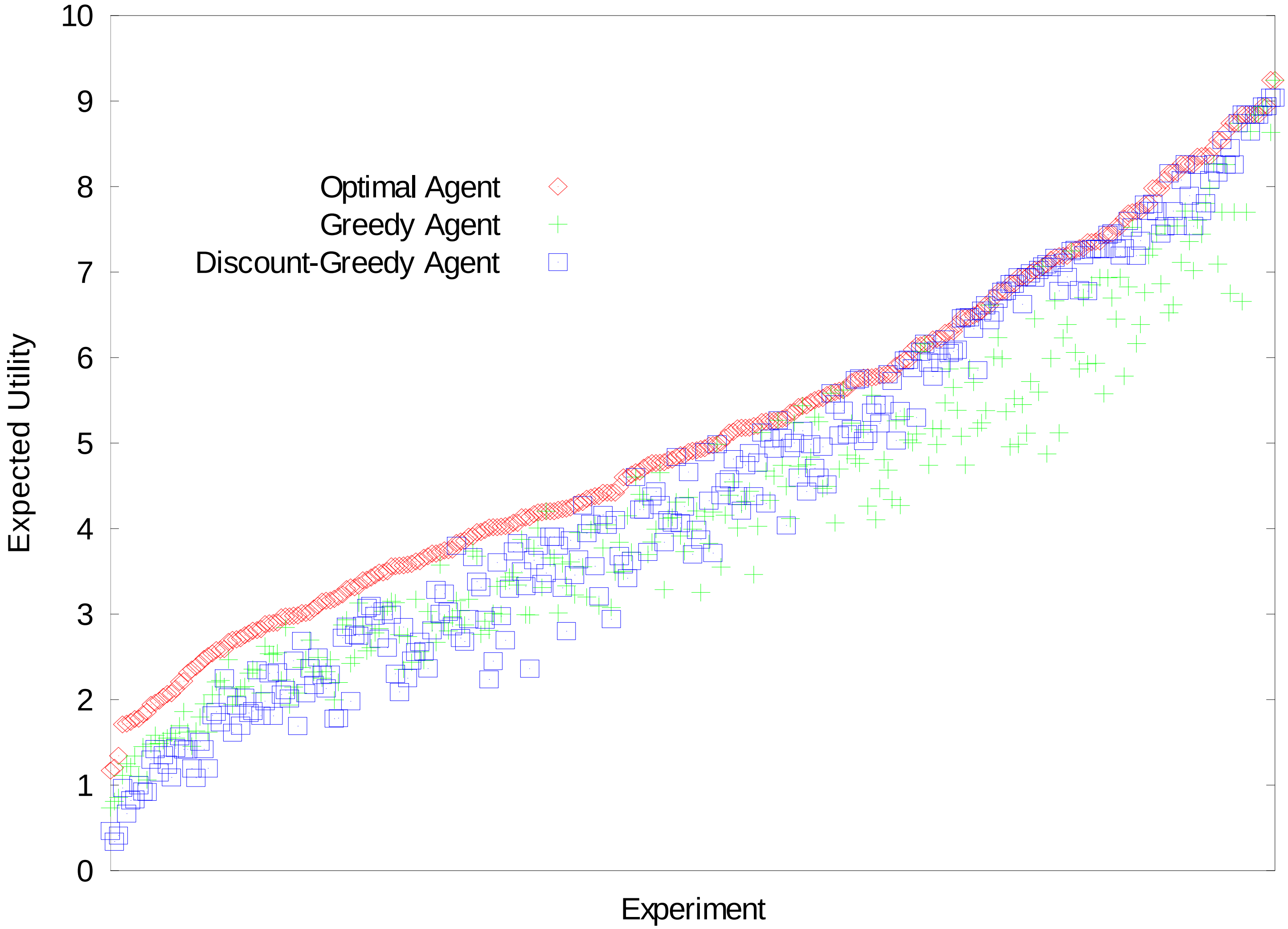
- Optimal ***static schedule*** would assign all future deliberation time to maximize expected mission utility.
- Inputs:
  - Phases with expected durations.
  - Problem configuration modification methods with expected utility descriptions (time/quality).
- Output:
  - Schedule assigning CSM methods to specific mission phases for all future deliberation time.

- Optimal ***policy*** would account for all nondeterministic outcomes of deliberation and world state.

Bounded-Horizon Discrete Schedule

- Only assign limited future deliberation time, in discretized intervals, to maximize expected utility of deliberation.
- Execute one or more of the scheduled deliberation activities (CSM methods) and then re-derive schedule.
  - Ala model predictive control.
- Greedy approach reduces complexity of deliberation scheduling.
- Reacts effectively to actual outcome of CSM processing.

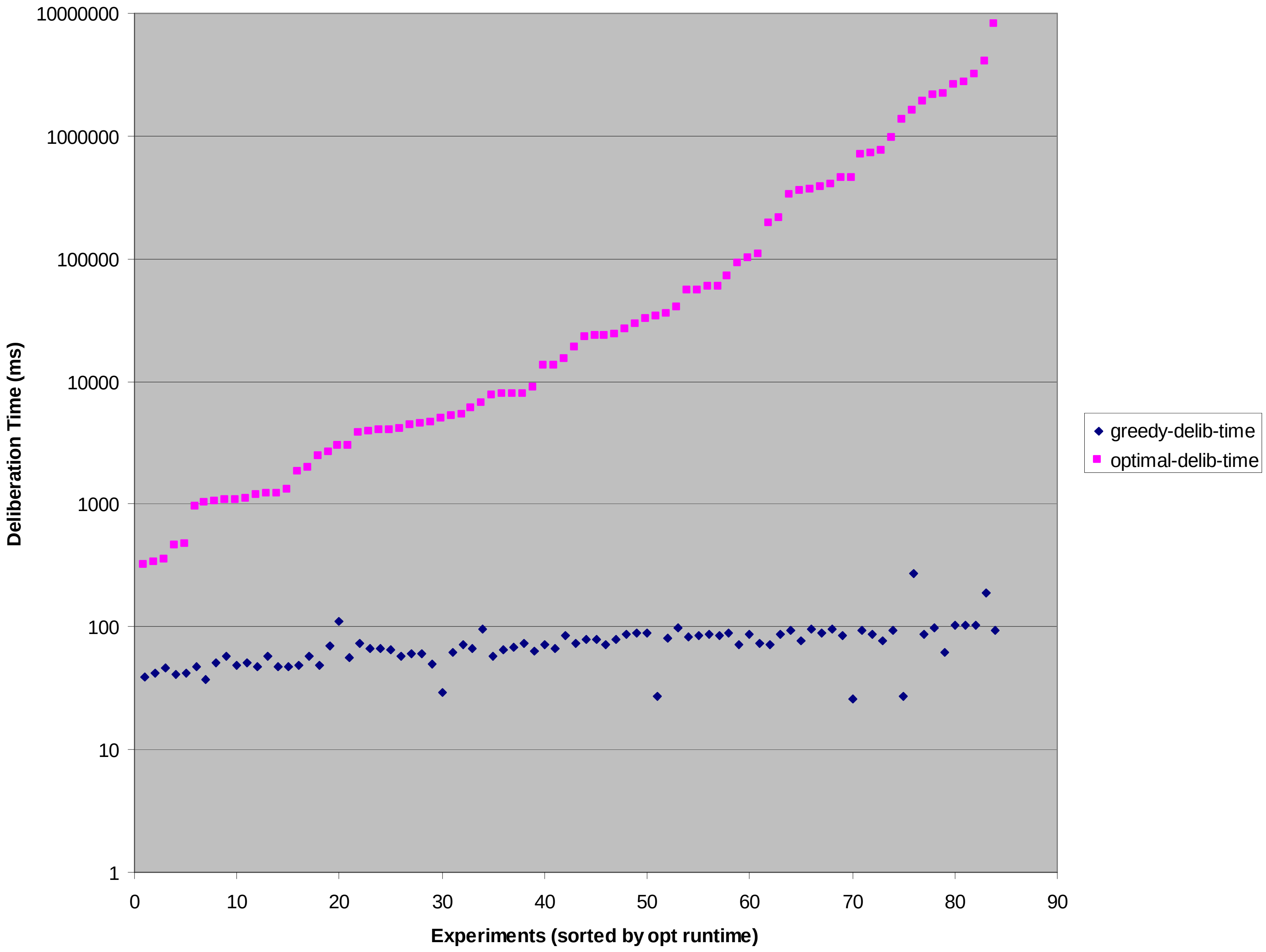
Comparing Deliberation Strategies: Results



Note: this is worst-case result on pathological scenarios.

SIFT

Runtime Comparison of Optimal & Greedy



Multi-Agent CIRCA

- CIRCA agents form teams and cooperatively plan for their team objectives.
- Coordinated plans can include real-time collaboration between agents.
- Negotiation protocols allow agents to restrict planning by better understanding teammates' expected behaviors.

- Meta-level control balances how much planning effort is spent for different mission phases, threats, and goals.

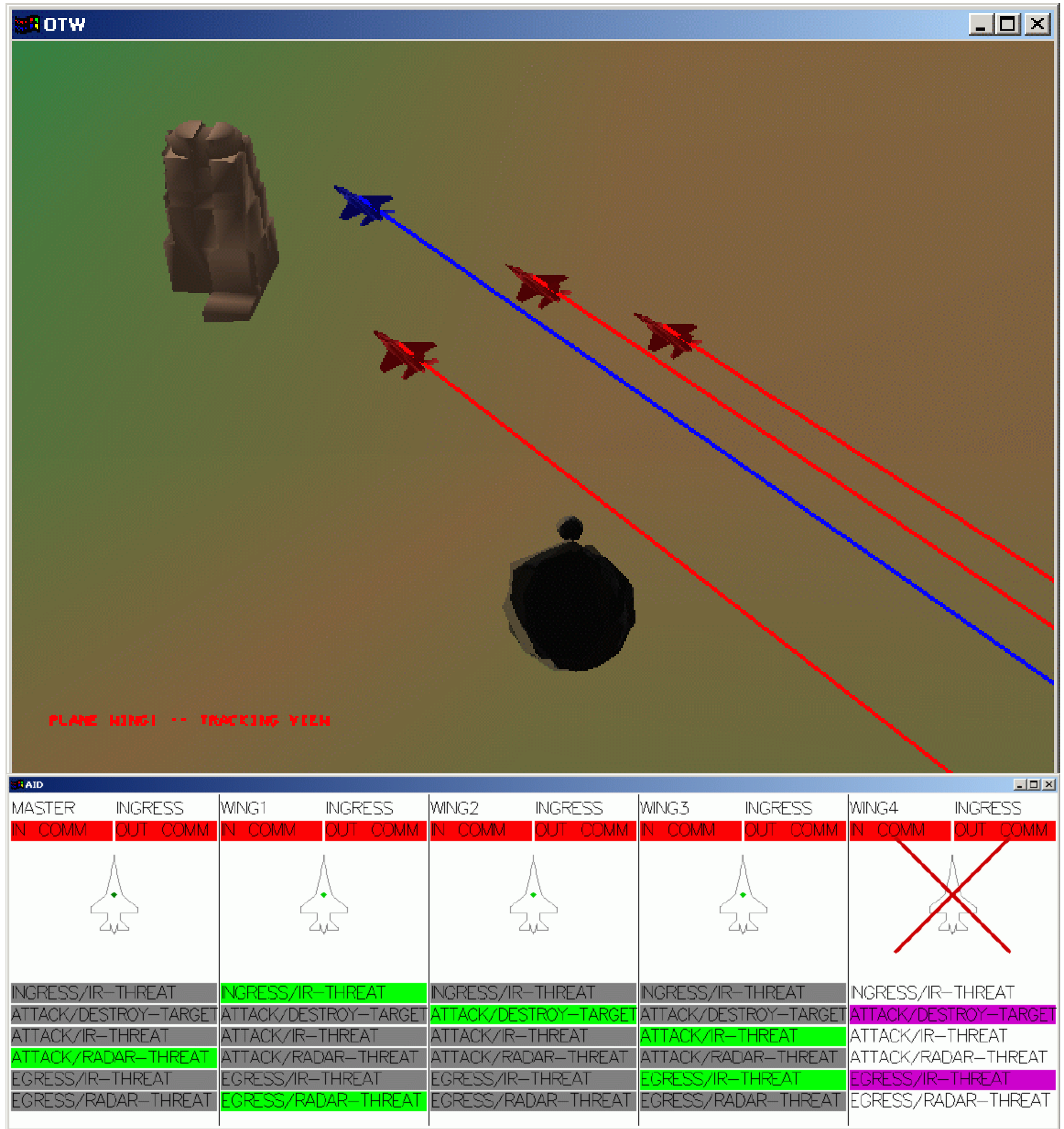
Multi-Agent Demo

- “Alert 5 scenario”: Build the best possible mission plan on the ground in limited time.
- Improve on the fly.
- Show how team of CIRCA agents reacts to popup threats.
- Show how team of CIRCA agents reacts to loss of assets.
- Illustrate corresponding deliberation scheduling problems and results.

- Quality meters illustrate status of plans for different mission phases, and deliberation scheduling decisions about focus of attention/replanning.

MASA-CIRCA Demonstration 1

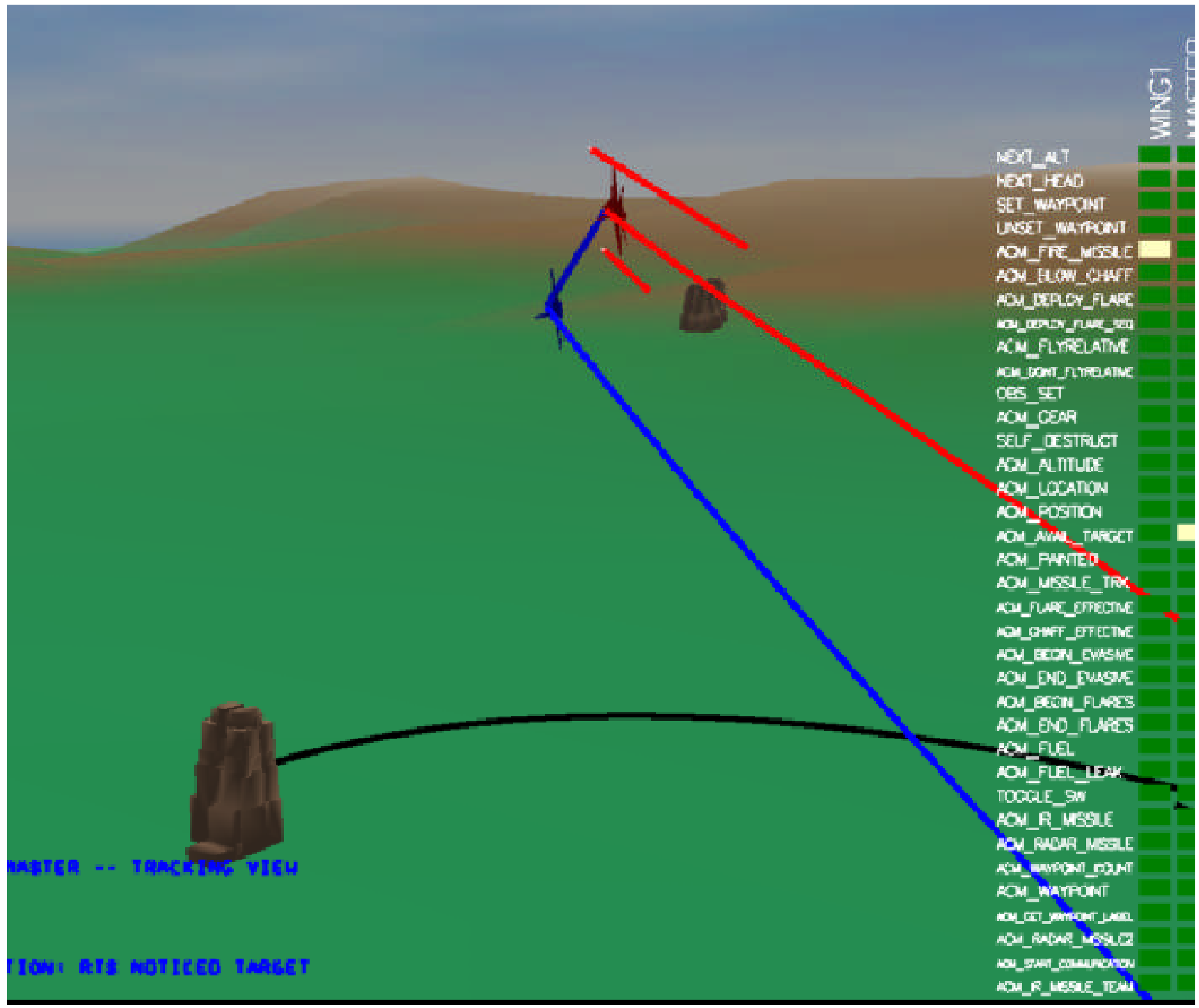
- Negotiation and dynamic renegotiation of roles and responsibilities.
- Dynamic replanning for changing missions.



SIFT

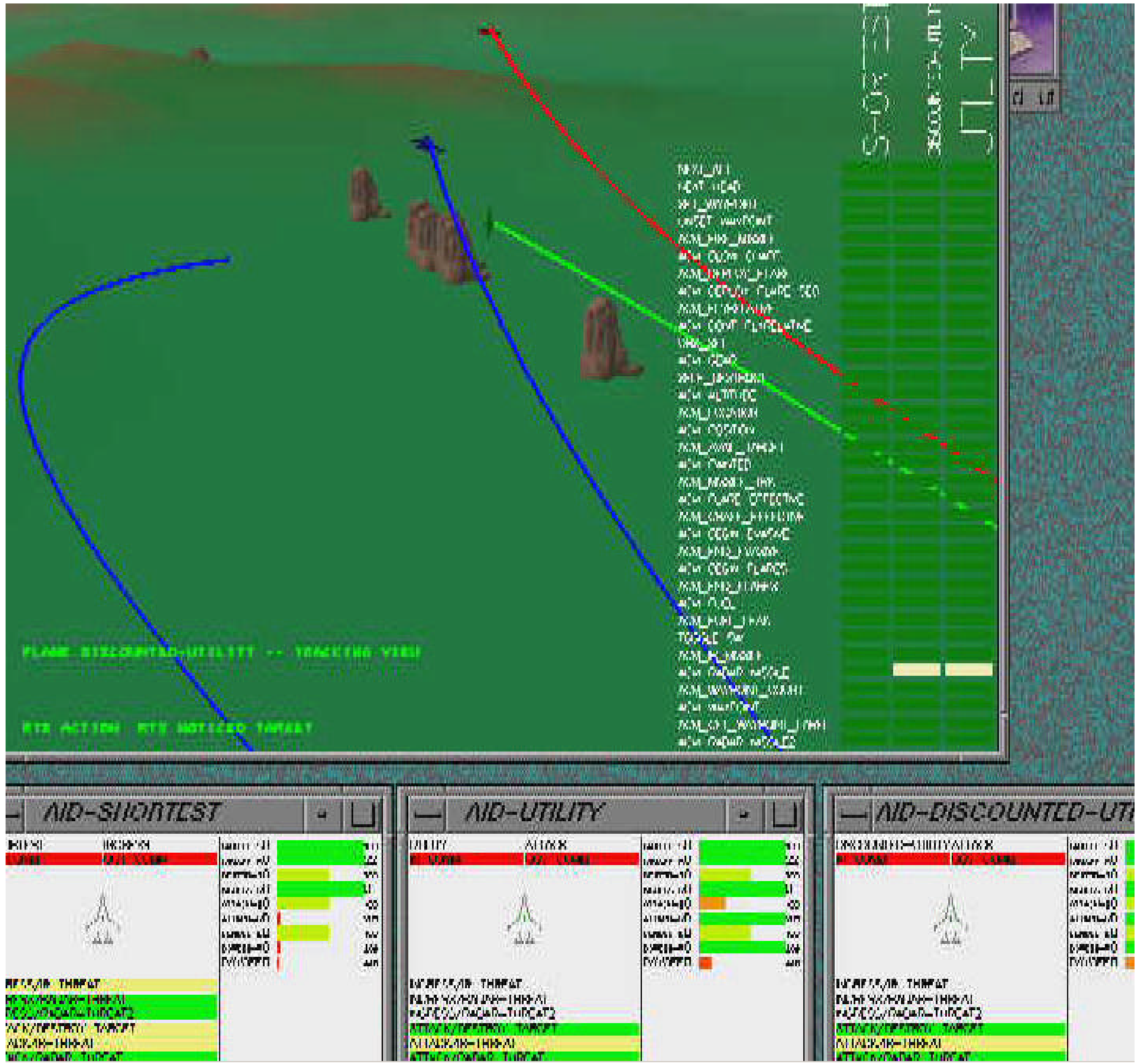
MASA-CIRCA Demonstration 2

- Planning for coordinated missions.
- Coordinated multi-aircraft mission execution.



MASA-CIRCA Demonstration 3

- Deliberation scheduling to explicitly manage planning effort.
- Comparison of different deliberation scheduling heuristics.



Provably Safe Plans are Not Always Good

- In early 2000, flying UAV simulations with CIRCA.
  - Threat-response to defeat inbound missiles.
  - Multi-vehicle coordinated behaviors (designate/shoot).
  - Adaptation to system failures and asset loss.

- But then… we told it the landing gear might fail.
- Planner returned very quickly with a safe plan:
- “Don’t take off”.

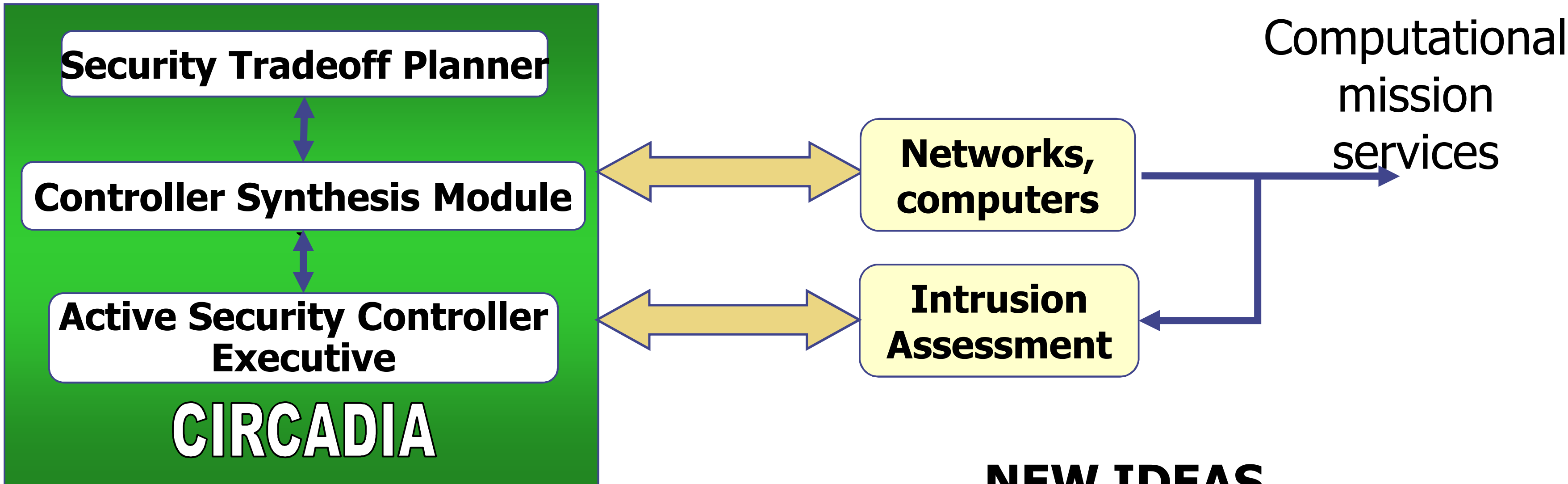
- Planner sacrificed all mission goals to remain safe.

- Some risk may be necessary.

Cyber-Security: Nothing is Certain

- In the ‘90s, cyber-attacks moved from hackers manually typing commands to autonomous viruses and scripted attacks.
- The tempo of cyber-war accelerated to near light speed.
- CIRCA to the rescue! CIRCADIA.
- Anticipate attacks, pre-position defensive responses and adapt cyber-defense posture to threat environment.
- Problem: can never be sure to defeat the threat.
  - No real lower bound on delay before attack succeeds.
  - Defenses may not always succeed.

CIRCADIA: Automatically Synthesizing Security Control Systems



IMPACT

- Automatic responses guaranteed to defeat intruders in real-time.
  - System derives appropriate responses for novel attack combinations.
- Automatic tradeoffs of security and monitoring vs. service and accessibility.

NEW IDEAS

- Use control theory to derive appropriate response actions automatically.
- Automatically tailor monitoring and responses according to mission, available resources, varying threats, and policies.
- Reason explicitly about response time requirements to provide performance guarantees.

SIFT

Probabilistic Controller Synthesis

- Problem: perfect control plan may not be possible.
- One approach: ignore less-likely situations.
- Add transition probabilities to state model.
  - World transitions and controlled actions.
- Build controllers that handle most-probable states.

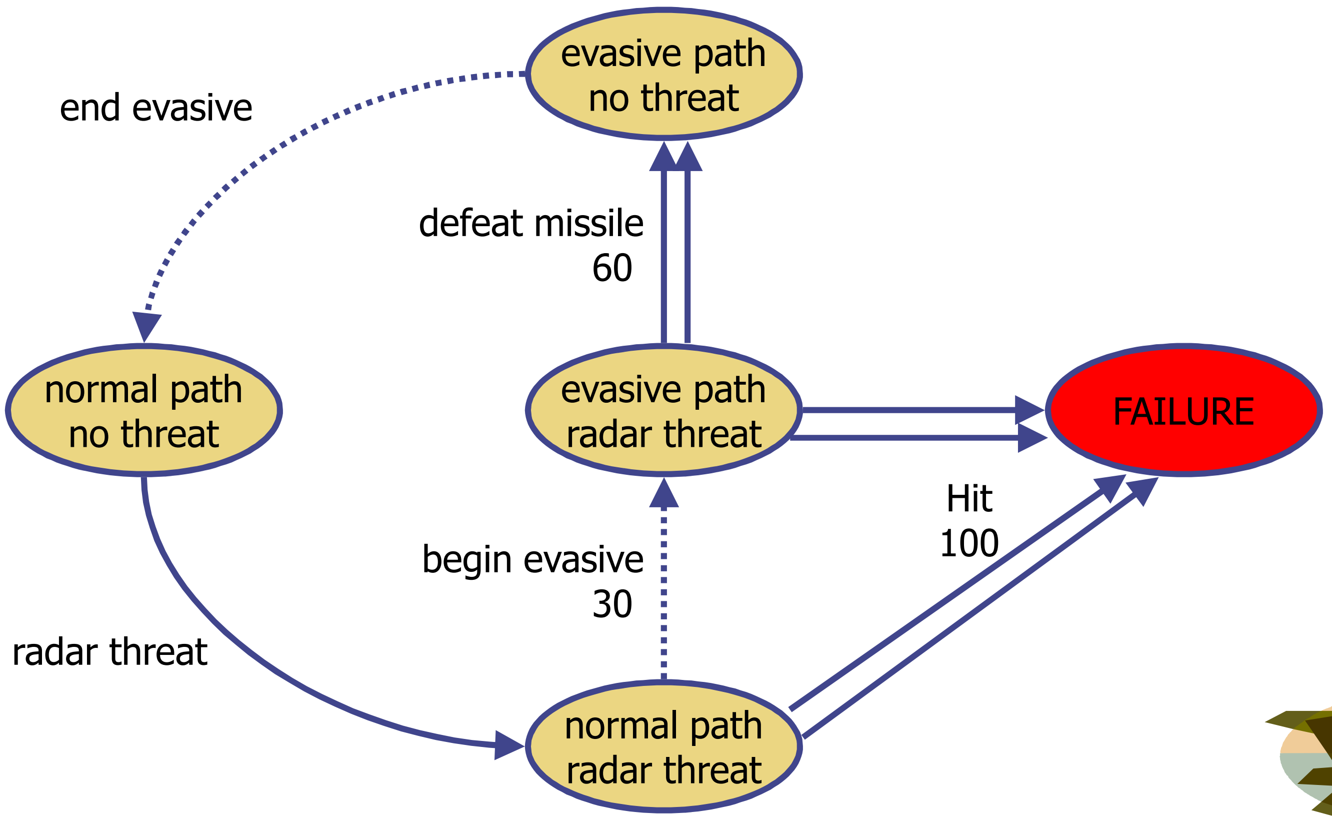
- Allows CIRCADIA to plan for imperfect, inherently unsafe situations.
- Also: trade off planning time and controller complexity against system safety.

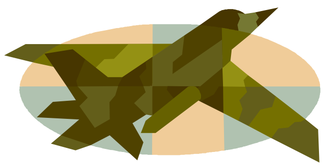
Probabilistic Transition Effects

- In the classical CIRCA framework, a transition can have nondeterministic transition time and nondeterministic effects.
- MEU mode adds transitions with *probabilistic transition time* and *probabilistic postconditions*:
  - Each transition has a probability distribution for transition time (*T-distribution*), and another distribution for the postconditions of the transition (*P-distribution*).
  - For each state, a set of transitions compete to trigger. The one with the shortest transition time (sampled from its T-distribution) wins and triggers the state transition.
  - Given a transition that won the trigger race, the next state is determined by sampling from its P-distribution.

Classic CIRCA World Model

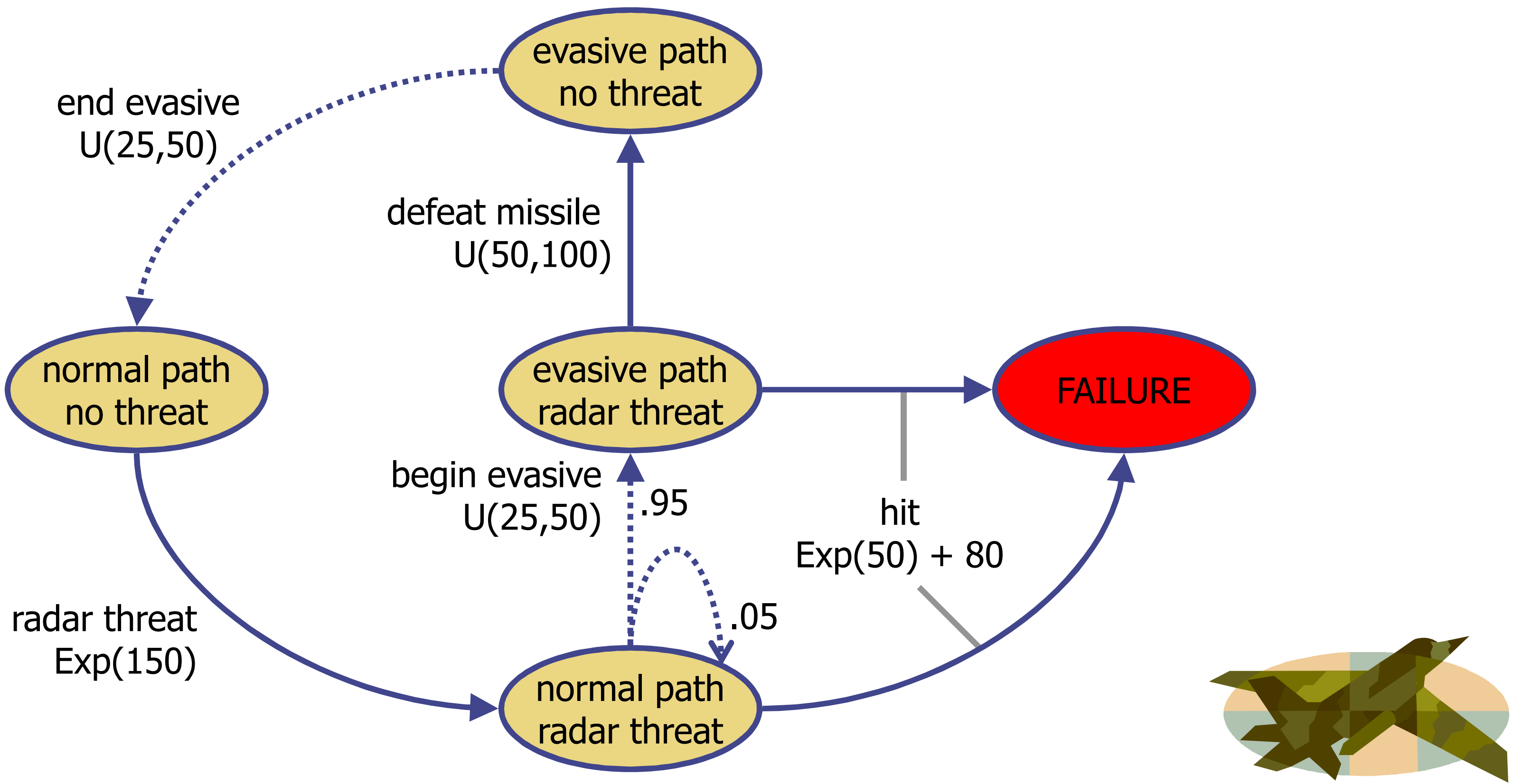
- A *plan* (or *controller*) chooses actions for states.
- Nonvolitional transitions can also move between states.
- Actions must be planned to *preempt* failure.





Probabilistic CIRCA World Model

- Time bounds can be distributions.
- Transitions can have probabilistic postconditions.

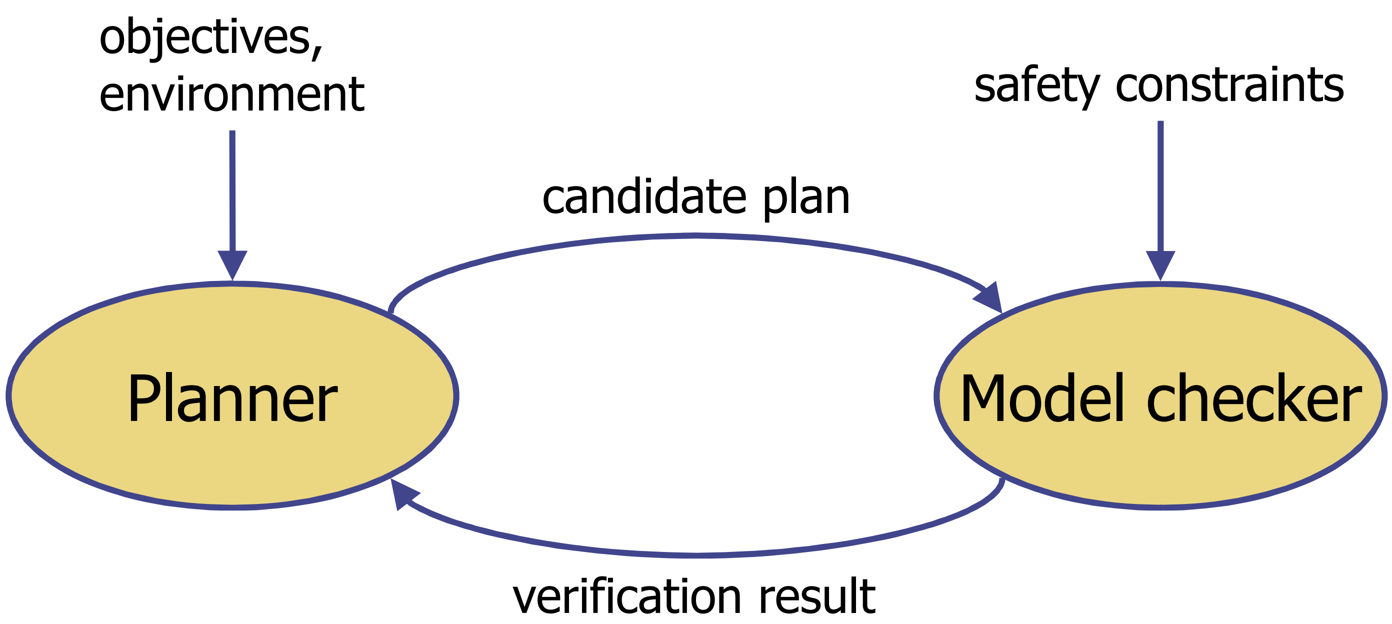


World Model Dynamics

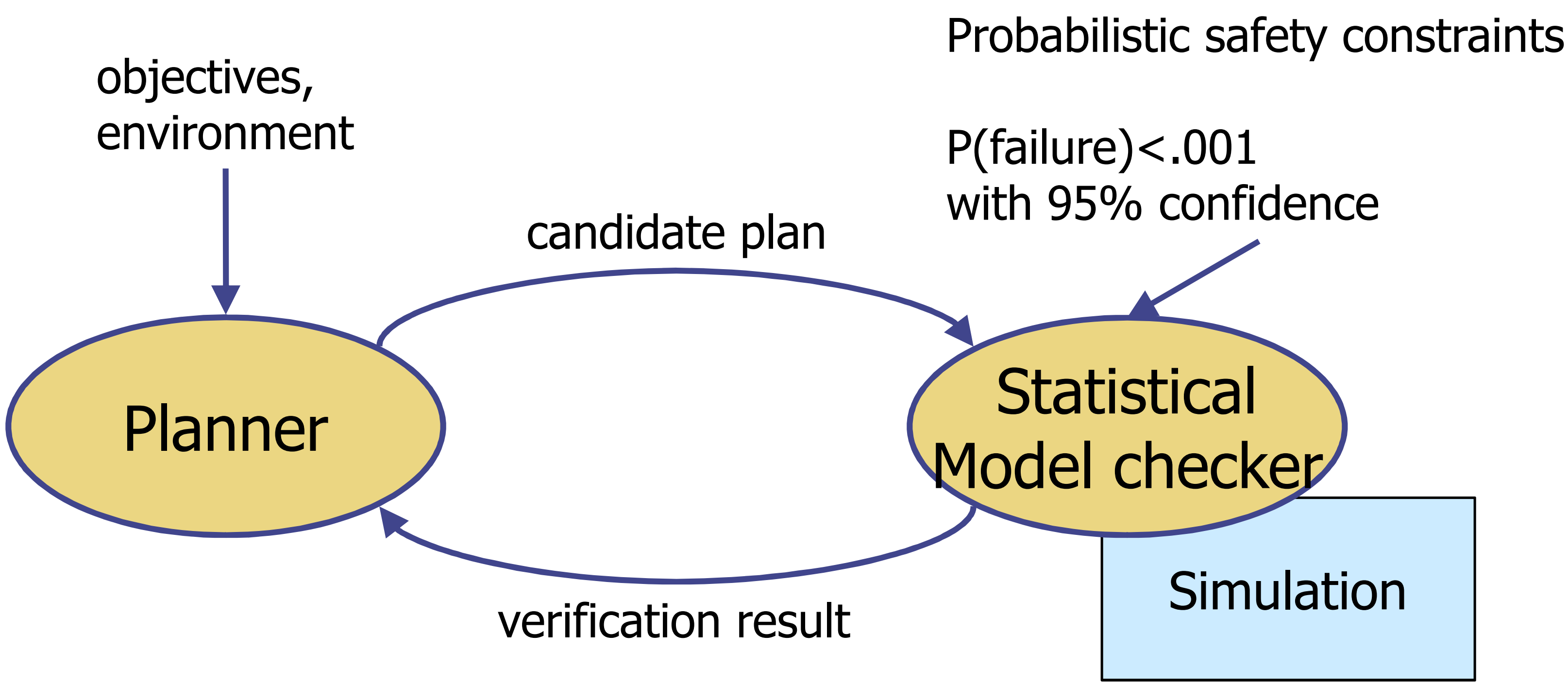
- The world model is a generalized semi-Markov process (GSMP).
- The world occupies a single state at any point in time.
- Enabled transitions in the current state compete to trigger.
- One transition triggers in each state, determining the next state.
- Non-Markovian because trigger distributions depend on dwell times.

- There are no analytic solutions for unrestricted GSMPs.
- Must use a sampling-based approach to estimate state probabilities (or determine if failure is too likely).
  - Build a plan, run sample executions to see if it is safe.

Planning with Model Checking



Planning with Statistical Model Checking

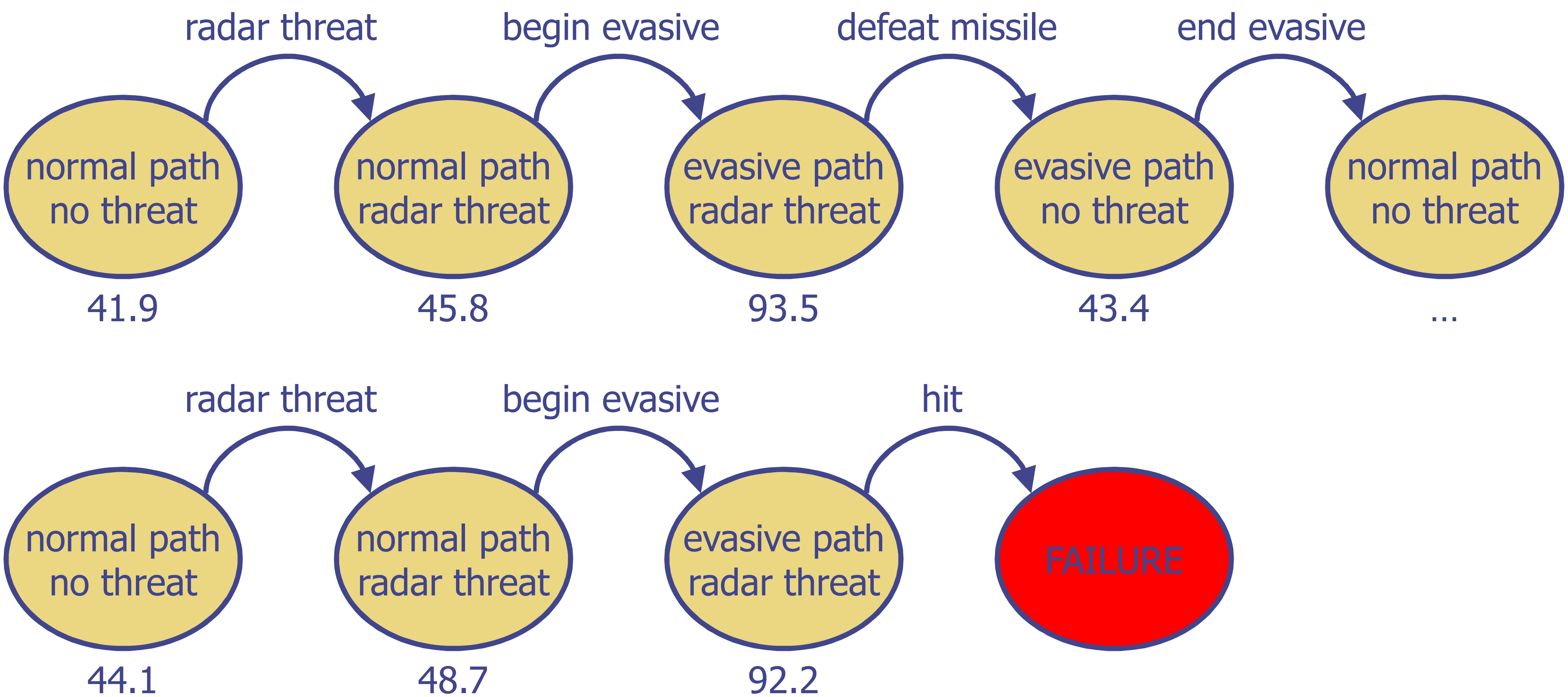


Probabilistic Controller Synthesis: MCSSP

- It may not be possible to guarantee 100% safety in realistic world models.
  - Time distributions, rather than fixed values.
  - Probabilistic outcomes, rather than always succeeding.

- Still we'd like to make plans that are very likely to keep the world safe and achieve goals.
- The Probabilistic CIRCA world model is a generalized semi-Markov process (GSMP).
- There are no analytic solutions for unrestricted GSMPs.
- Must use a sampling-based approach to estimate state probabilities (e.g., to determine if failure is too likely).

Sample Execution Paths



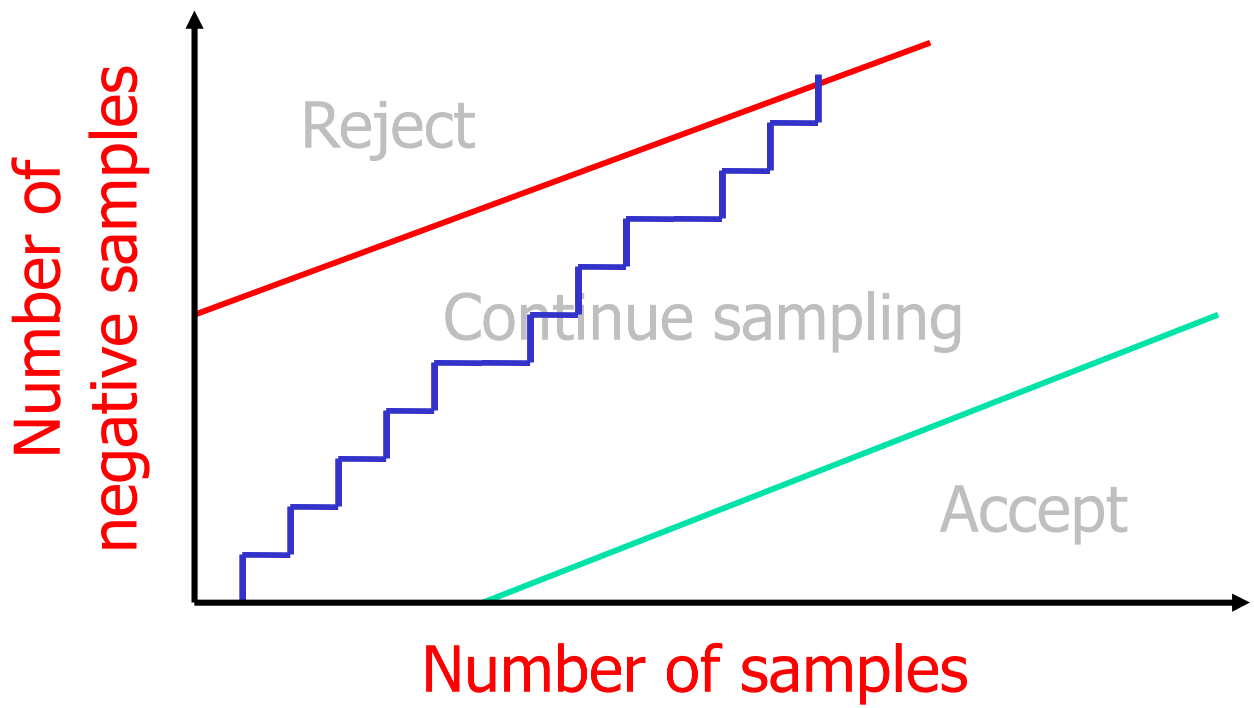
Plan Safety

- Two parameters:
  - Failure probability threshold: θ.
  - Maximum execution time (horizon): t_{max}.
- A plan is safe if the probability of reaching a failure state within t_{max} time units is at most θ.

SIFT

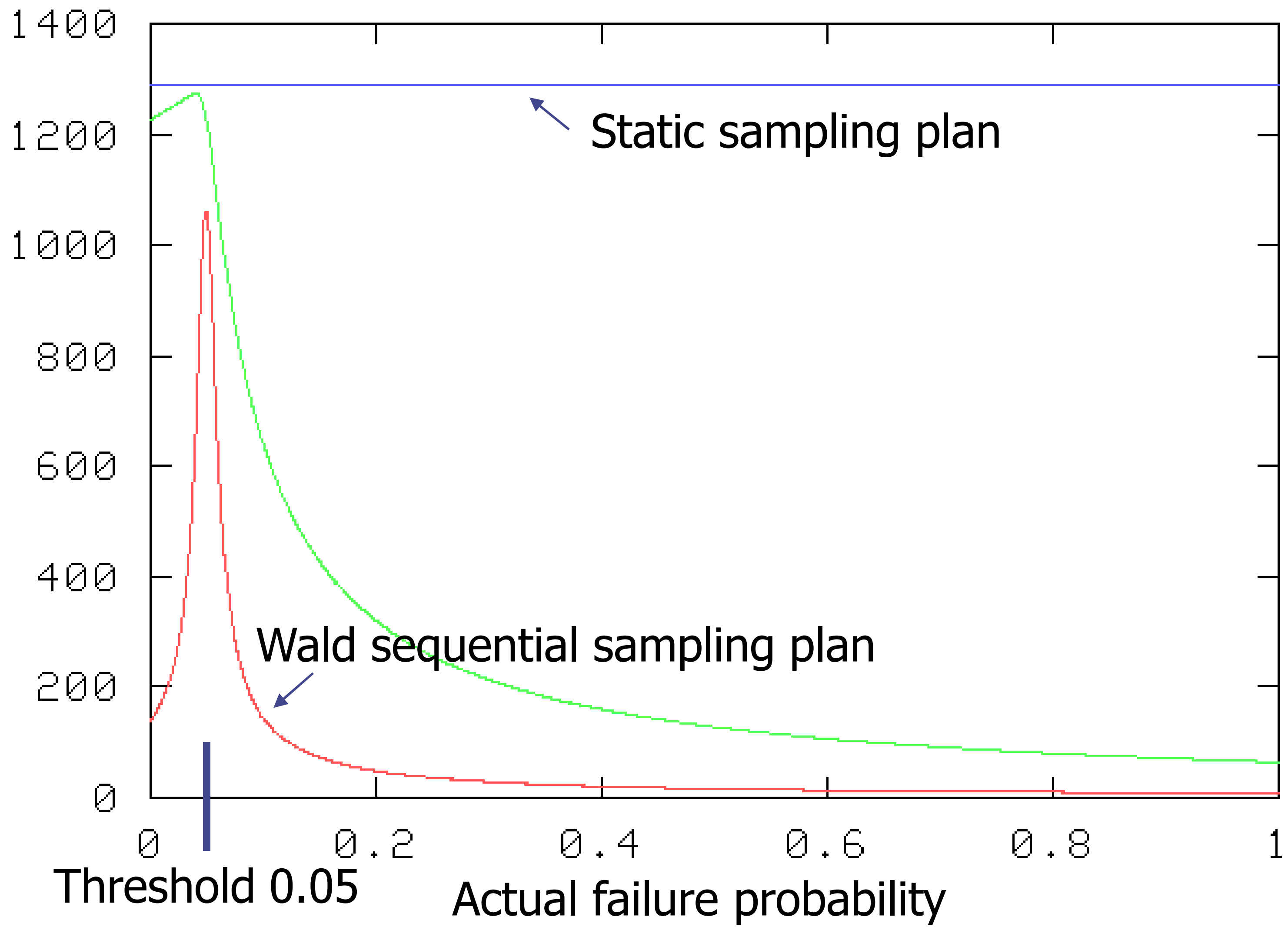
Graphical Representation of Sequential Test

- Reject hypothesis

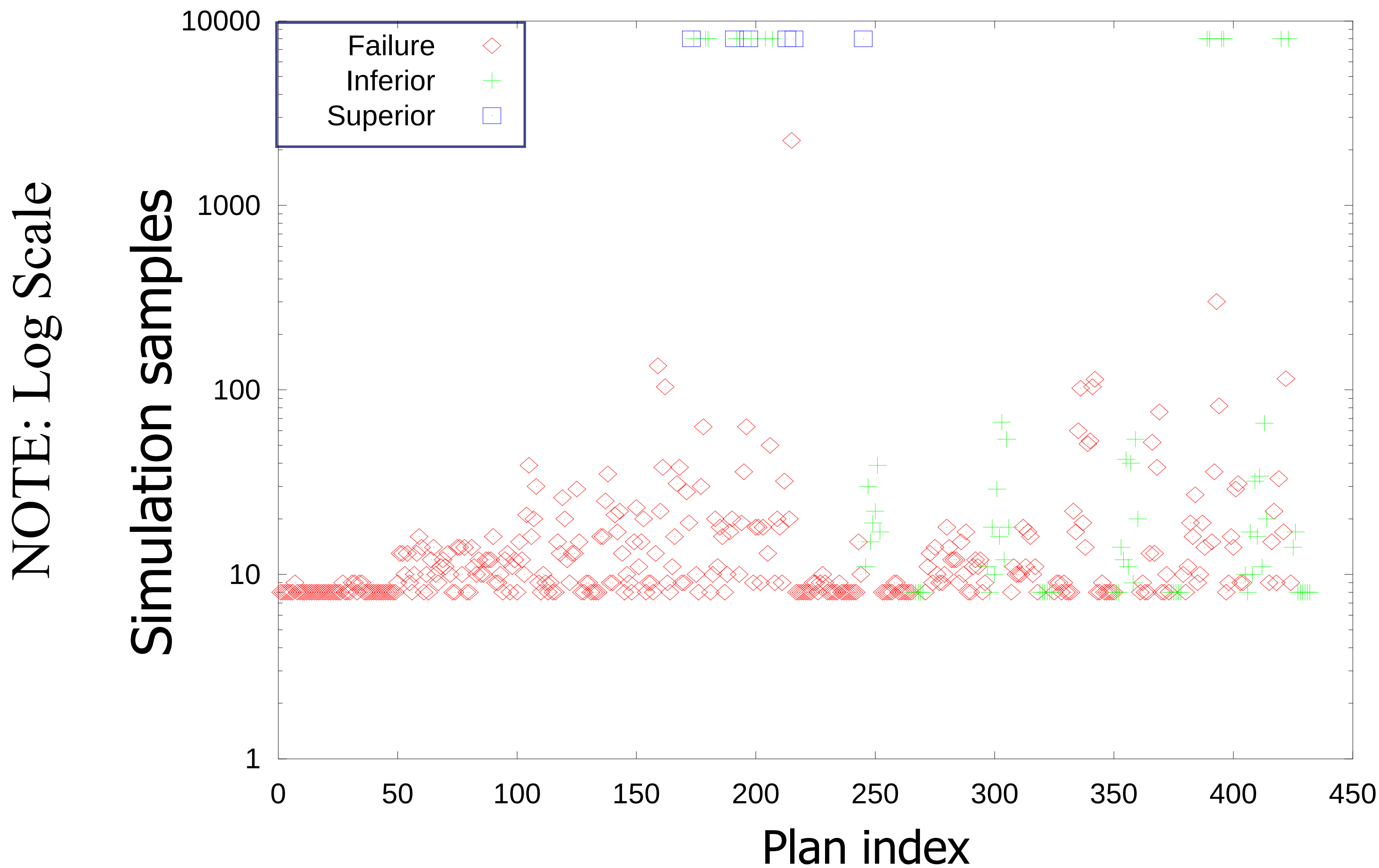


Number of Samples Required

Wald acceptance sampling requires significantly fewer samples.



Sequential Approach Avoids 95% of Sampling



ε=.025

δ=.05

- Static sampling plan would require 8000 samples per plan.
  - Sequential sampling averages 372 samples per plan.

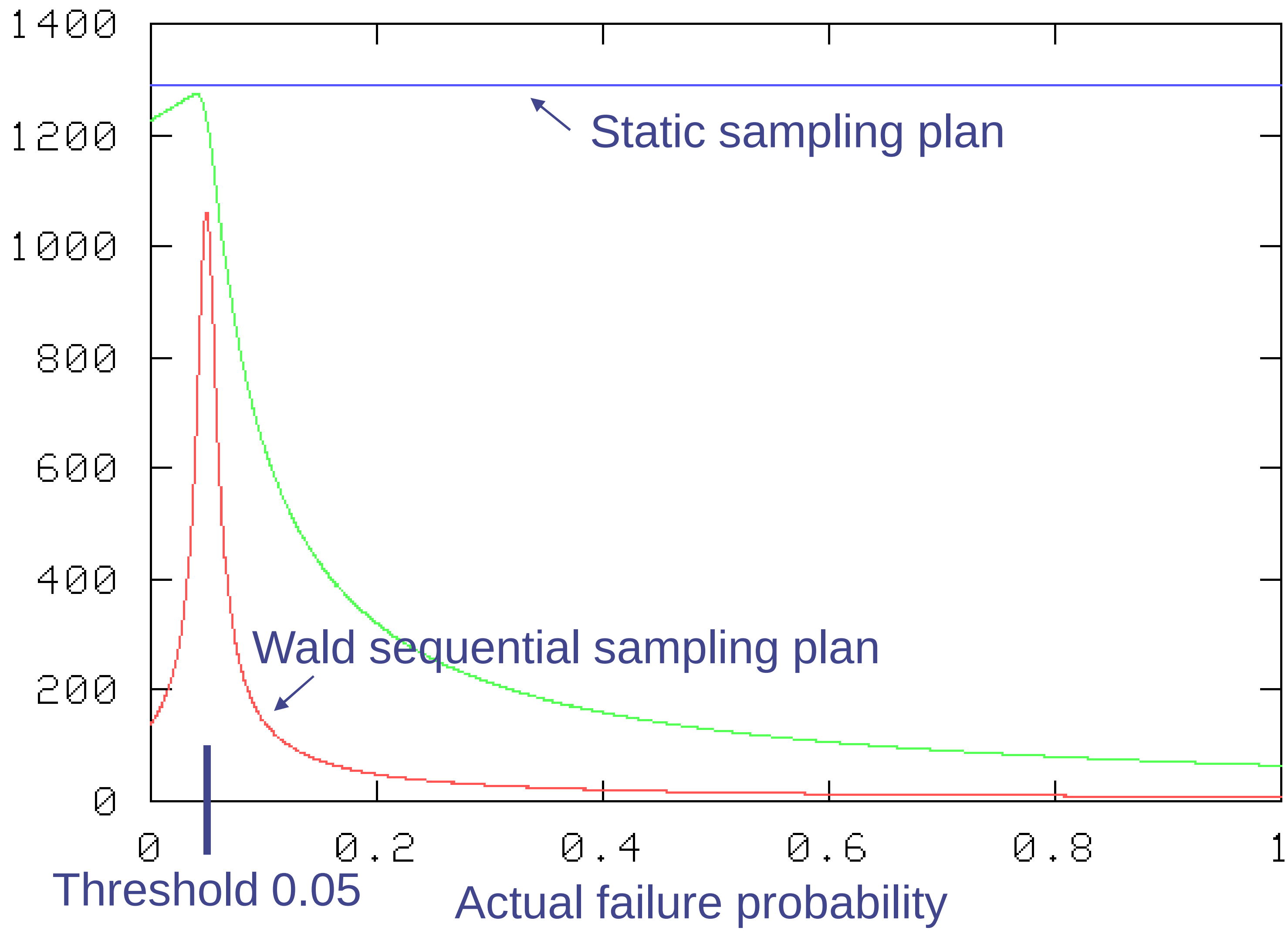
SIFT

New Decision-Theoretic CIRCA Planning

- Decision theory provides mechanism to trade risk against goal achievement using expected utility.
- Add a reward model to capture relative value of mission goals and inherent costs of security actions.
- Build candidate plans.
- Run simulation samples to estimate expected utility.
- Iterate and save best plan until run out of planning time or there are no more plans.
- Maximize Expected Utility (MEU) mode.

Number of Samples Required

Wald acceptance sampling requires significantly fewer samples.



SIFT

Additional Topics

- Reward models added, so CIRCA can trade risk against reward: decision theory.
- Importance sampling investigated for very low probability events.
- More complex hybrid dynamics.

- Distributed multi-agent negotiation over roles and responsibilities in team missions.
  - Continuous monitoring and replanning: failure recovery including renegotiation.
- Deliberation scheduling / meta-control.

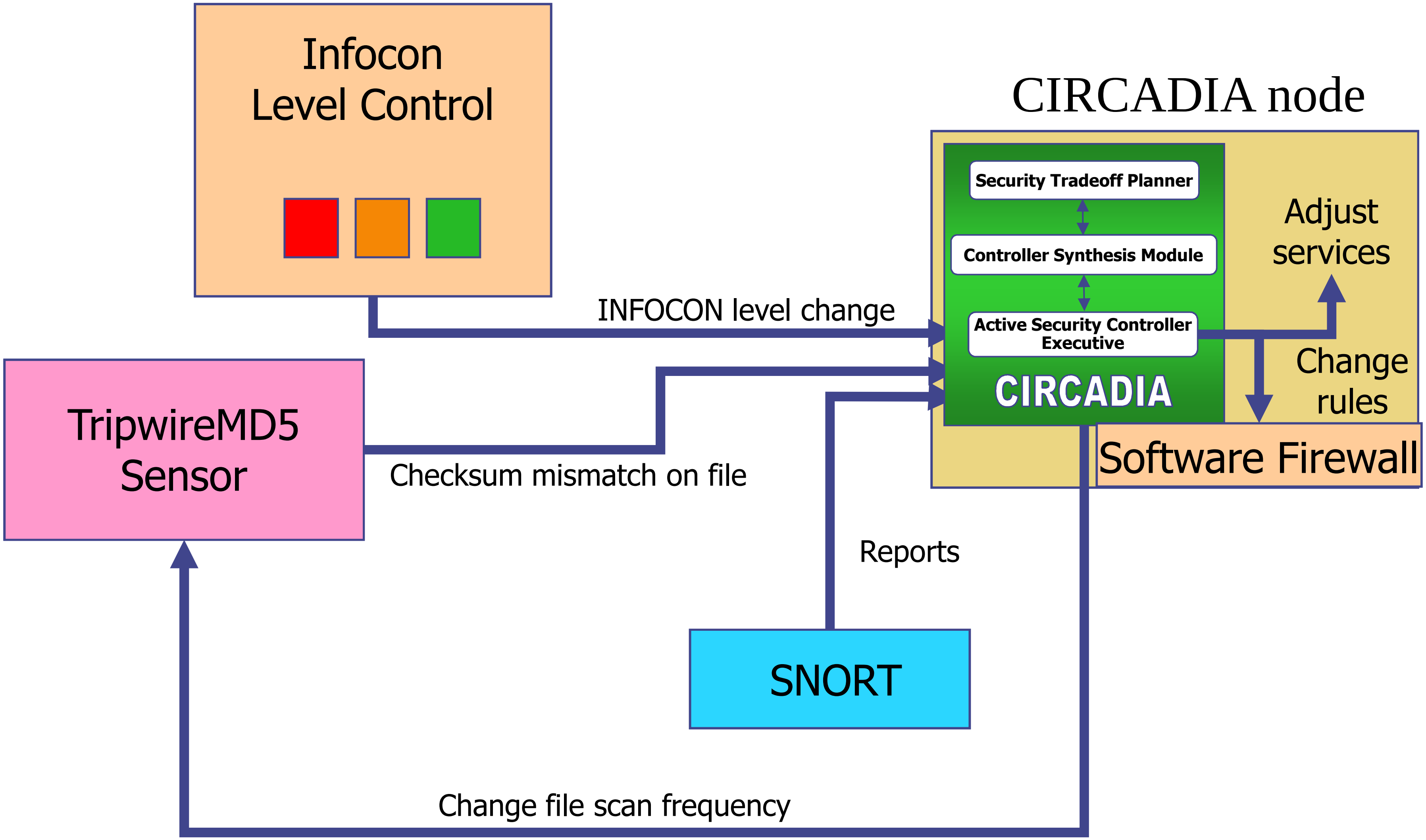
Reward for CIRCADIA Models

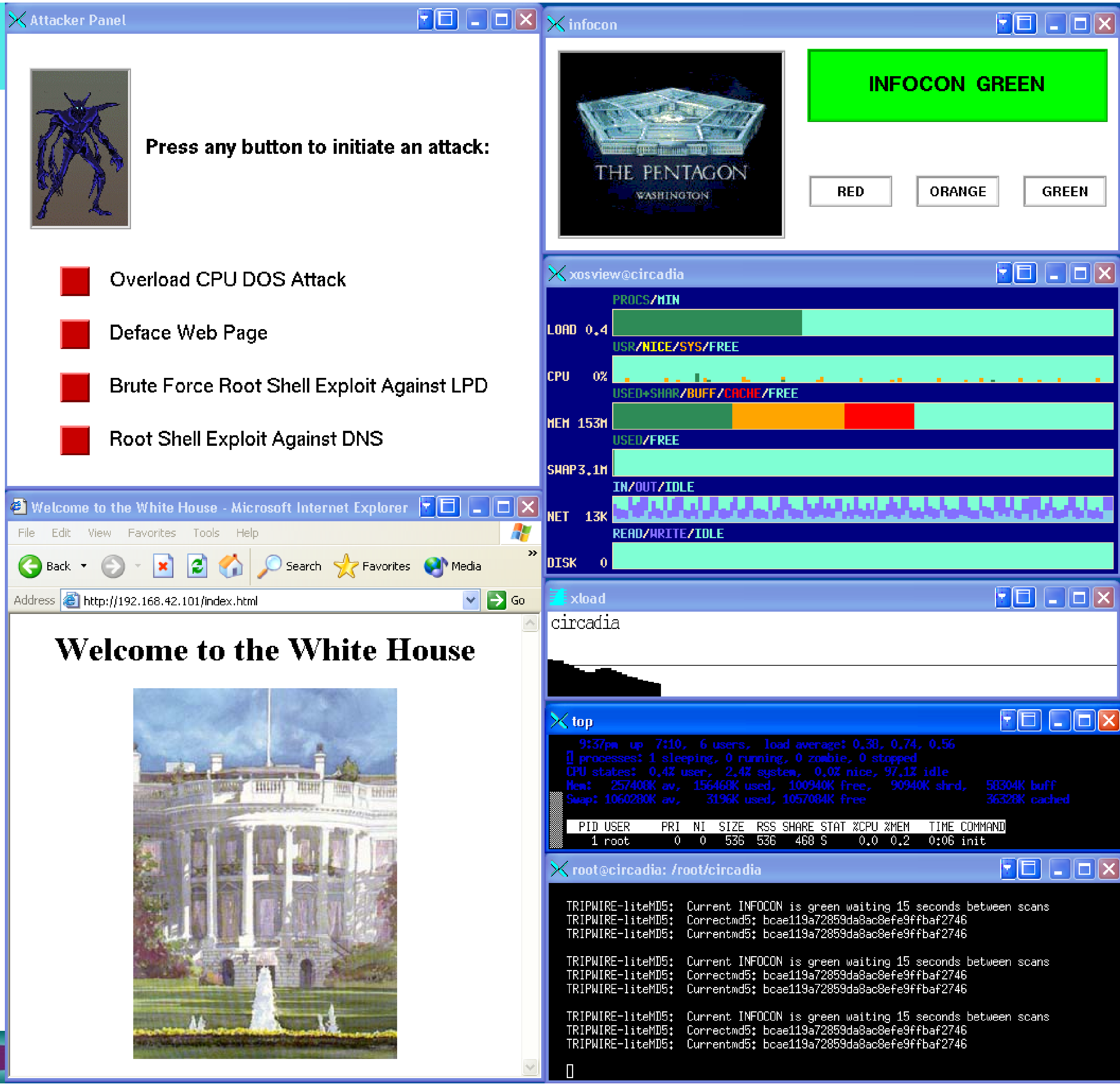
- Maintenance/accumulation goals: more value the longer you stay in those states ("Web server is up").
  - Using dwell-time-weighted probability.
- Repeated achievement/reaction goals (opportunistic): get value each time you achieve ("Sanitize compromised machines").
- One-shot achievement goals: get all the value as soon as you get there ("Complete network self-configuration").
- Cost of actions and losses/failure ("Attacker compromises database" vs. "Attacker gains root").
- Overall utility is sum of these rewards $U = U_M + U_A + U_{RA} - U_{cost.}$

Estimating a Plan's Expected Utility

- Due to the complexity of the goal models and non-Markovian time representation, the EU is difficult to compute analytically.
- Thus we turn to the *sampling-based* approach.
- What is the purpose of sampling? Not necessarily to estimate the EU!
- We can sample to:
  - Determine if the current plan is *too likely to fail* (*hypothesis testing*).
  - Determine if the current plan has *lower EU than the current best plan* (*hypothesis testing*).
  - Estimate the EU of the current plan to with *given error margin and given confidence coefficient* (*interval estimation*).
- All of these can be done *sequentially* (which saves time).

Functions on CIRCADIA Node





Attacker Panel
Press any button to initiate an attack:
Overload CPU DOS Attack
Deface Web Page
Brute Force Root Shell Exploit Against LPD
Root Shell Exploit Against DNS
infocon
THE PENTAGON
WASHINGTON
INFOCON GREEN
RED
ORANGE
GREEN
xosview@circadia
PROCS/MIN
LOAD 0.4
USR/NICE/SYS/FREE
CPU 0%
USED+SHAR/BUFF/CACHE/FREE
MEM 153M
USED/FREE
SWAP3.1M
IN/OUT/IDLE
NET 13K
READ/WRITE/IDLE
DISK 0
Welcome to the White House - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://192.168.42.101/index.html
Go
Welcome to the White House
xload
circadia
top
9:37pm up 7:10, 6 users, load average: 0.38, 0.74, 0.56
1 processes: 1 sleeping, 0 running, 0 zombie, 0 stopped
CPU states: 0.4% user, 2.4% system, 0.0% nice, 97.1% idle
Mem: 257408K av, 156468K used, 100940K free, 90940K shrd, 58304K buff
Swap: 1060280K av, 3196K used, 1057084K free 36328K cached
PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
1 root 0 0 536 536 468 S 0.0 0.2 0:06 init
root@circadia: /root/circadia
TRIPWIRE-liteMD5: Current INFOCON is green waiting 15 seconds between scans
TRIPWIRE-liteMD5: Correctmd5: bcae119a72859da8ac8efe9ffbaf2746
TRIPWIRE-liteMD5: Currentmd5: bcae119a72859da8ac8efe9ffbaf2746
TRIPWIRE-liteMD5: Current INFOCON is green waiting 15 seconds between scans
TRIPWIRE-liteMD5: Correctmd5: bcae119a72859da8ac8efe9ffbaf2746
TRIPWIRE-liteMD5: Currentmd5: bcae119a72859da8ac8efe9ffbaf2746
TRIPWIRE-liteMD5: Current INFOCON is green waiting 15 seconds between scans
TRIPWIRE-liteMD5: Correctmd5: bcae119a72859da8ac8efe9ffbaf2746
TRIPWIRE-liteMD5: Currentmd5: bcae119a72859da8ac8efe9ffbaf2746

SIFT

Major CIRCA Innovations

- Proven-safe real-time closed-loop control plans, derived on the fly.
- Coordinated multi-agent plans with real-time guarantees ("you sense, I'll act").
- Incremental model checking for efficient plan verification (patented).
- Pruning of plan spaces based on failure probability.
- Coordination algorithms that share partial plan information to resolve over-constrained domains.
- Domain-independent heuristic methods to guide refinement and search.
- MDP modeling of deliberation scheduling algorithms.
- Deliberation scheduling implementation with explicit control over problem solver complexity.

MASA-CIRCA Demonstrated Capabilities

- Negotiation and dynamic renegotiation of roles and responsibilities.
- Dynamic replanning for changing missions.
- Planning for coordinated missions.
- Coordinated multi-aircraft mission execution.
- Deliberation scheduling to explicitly manage planning effort.

- Runtime plan execution monitoring to detect unexpected states.

After All This…

- We have better methods to reason about autonomous plans and behaviors.
- Scalability remains challenging, but some problems are within reach.

- The big challenges now:
  - Specifying what you really want.
  - Describing how the world really works.
  - Making people accept risky autonomy.
  - Brittleness…. What to do when the world doesn't behave as expected.   Learning?

Thank You for Your Attention

Questions?

SIFT

CIRCA: The Cooperative Intelligent Real-Time Control Architecture

- Planning for real-time reactions.
- Formal verification of plan safety.
- Real-time reactive plan execution.
- Active meta-control to manage planning/deliberation time.

- Applications:
  - Coordinated UAV teams.
  - Autonomous spacecraft.
  - Self-regenerative cyber security.