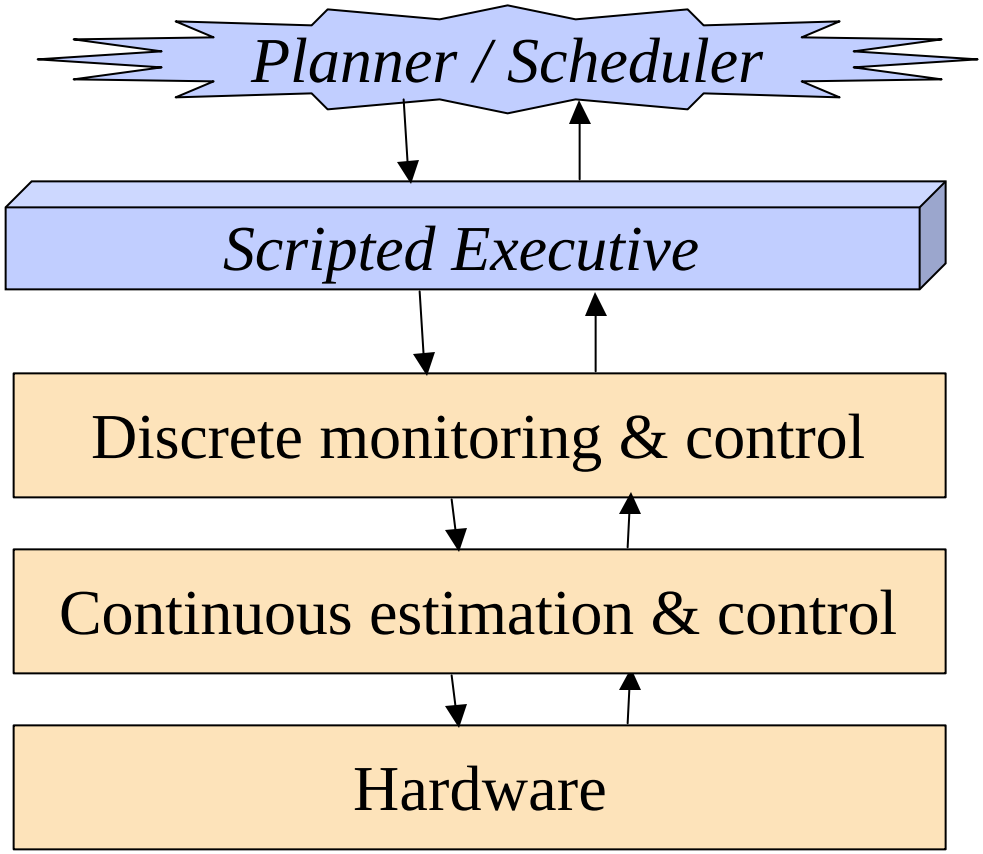
Model-based Autonomous Systems: An Overview

Programming Complex Autonomous Systems

Are Traditional Robot Control Architectures adequate?



typically eliminated

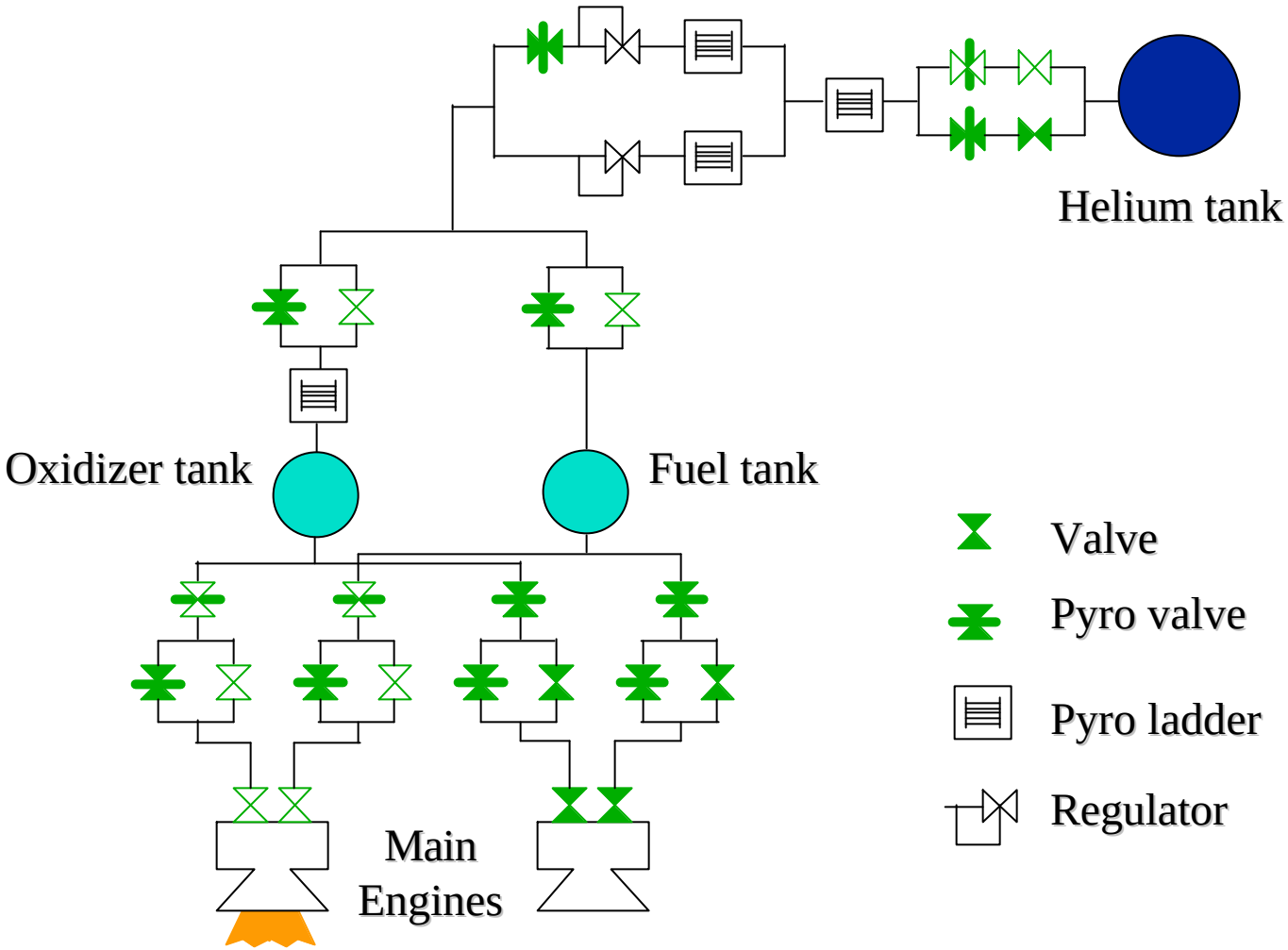
Challenge: Autonomous Saturn Orbital Insertion of Cassini

[Williams & Nayak 96a; Pell *et. al*. 97]

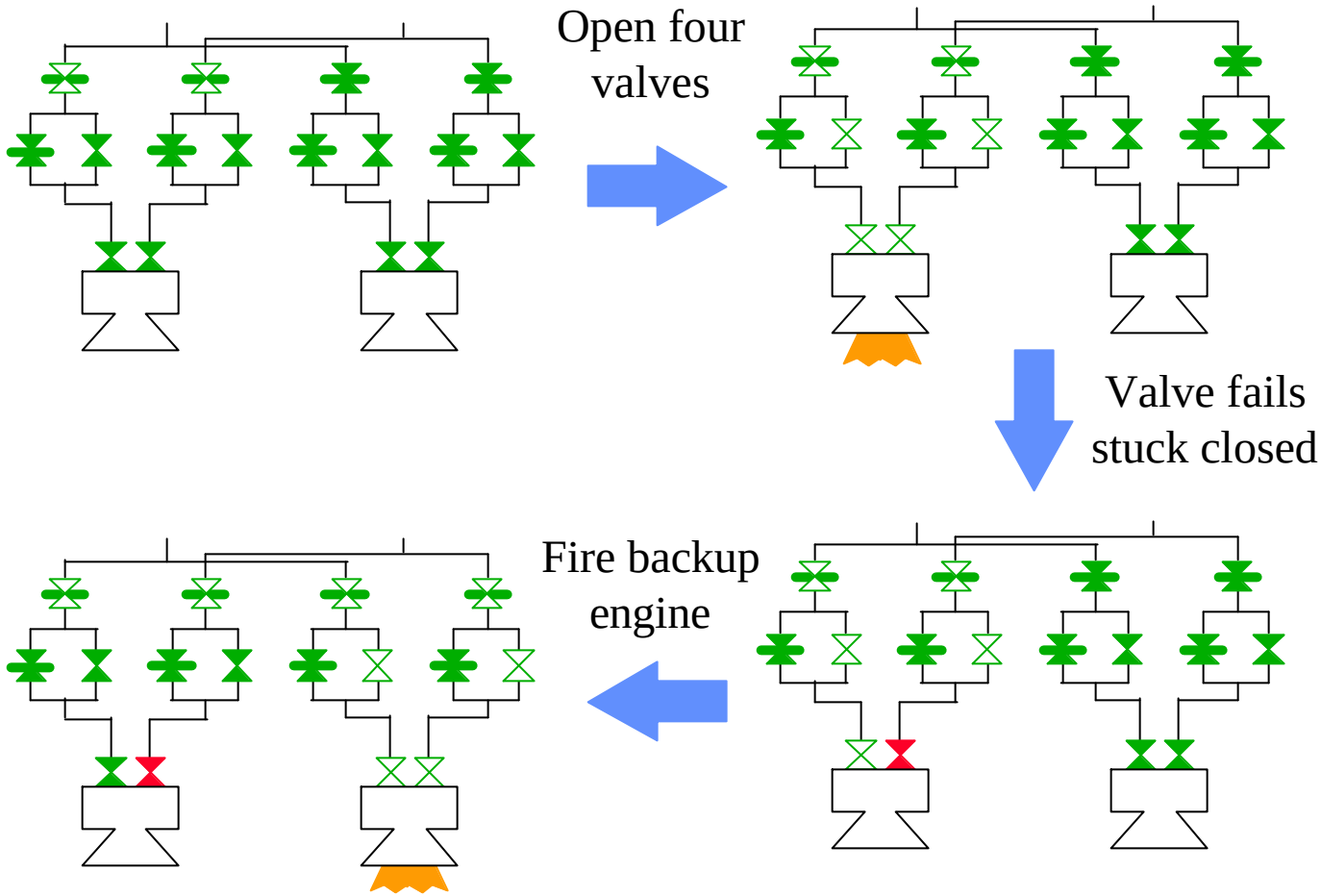
QuickTime™ and a
Photo - JPEG decompressor
are needed to see this picture

- *no Earth Comm*
- *~ 1 hr insertion window*
- *engines idle for several years*
- *moves through ring plane*

Cassini Propulsion System Schematic



Reconfiguring for a failed engine



Autonomous System Coding Challenge

Programmers must reason through system-wide interactions to generate codes for:

- monitoring
- hardware mode confirmation
- goal tracking
- detecting anomalies
- isolating faults
- diagnosing causes
- parameter estimation
- hardware reconfiguration
- fault recovery
- standby
- safing
- fault avoidance
- adaptive control
- control policy coordination

→ poor reuse, poor coverage, error prone

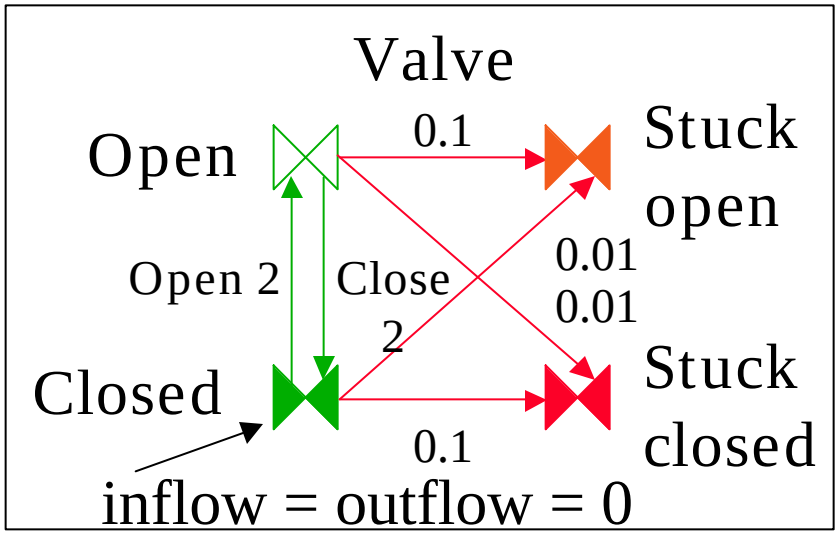
Solution: Part 1
Model-based Programming

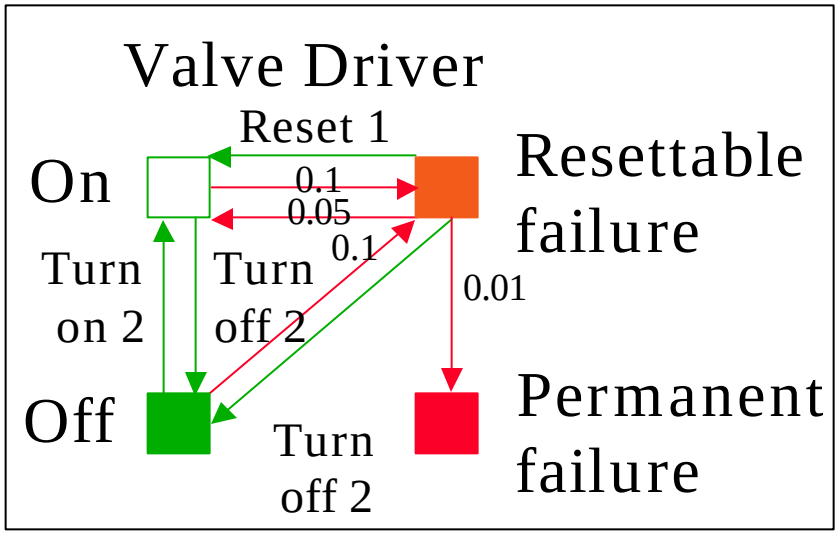
Programmers and operators generate breadth of functions from commonsense hardware models in light of mission-level goals.

Have engineers program in models, automate synthesis of code:

- models are compositional & highly reusable.
- generative approach covers broad set of behaviors.
- commonsense models are easy to articulate at concept stage and insensitive to design variations.

Single Core Model Describes Software and Hardware dynamics





Finite Constraint, Concurrent Probabilistic Transition Systems

- Models qualitative behavior of component modes
- Models failure, intermittency and optimal control

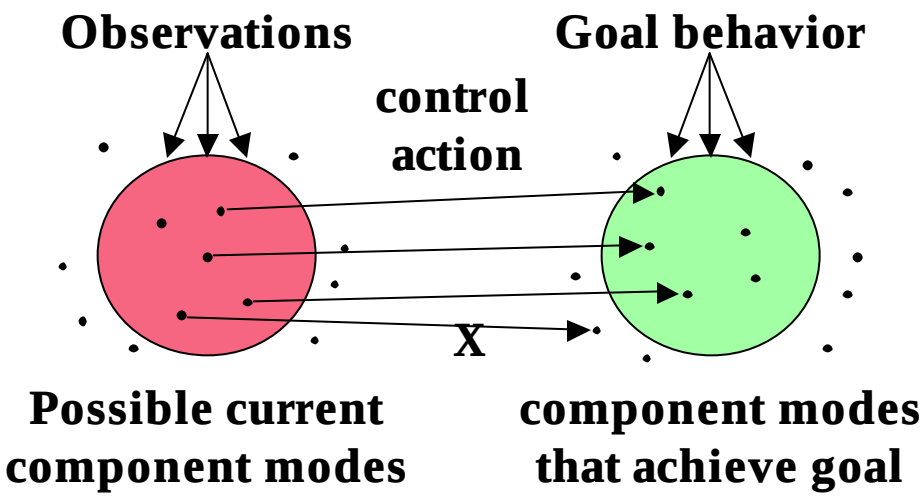
Autonomous System Code Identifies and Reconfigures Component Modes

Identifying Modes:

- mode confirmation
- goal tracking
- detecting anomalies
- isolating faults
- diagnosing causes

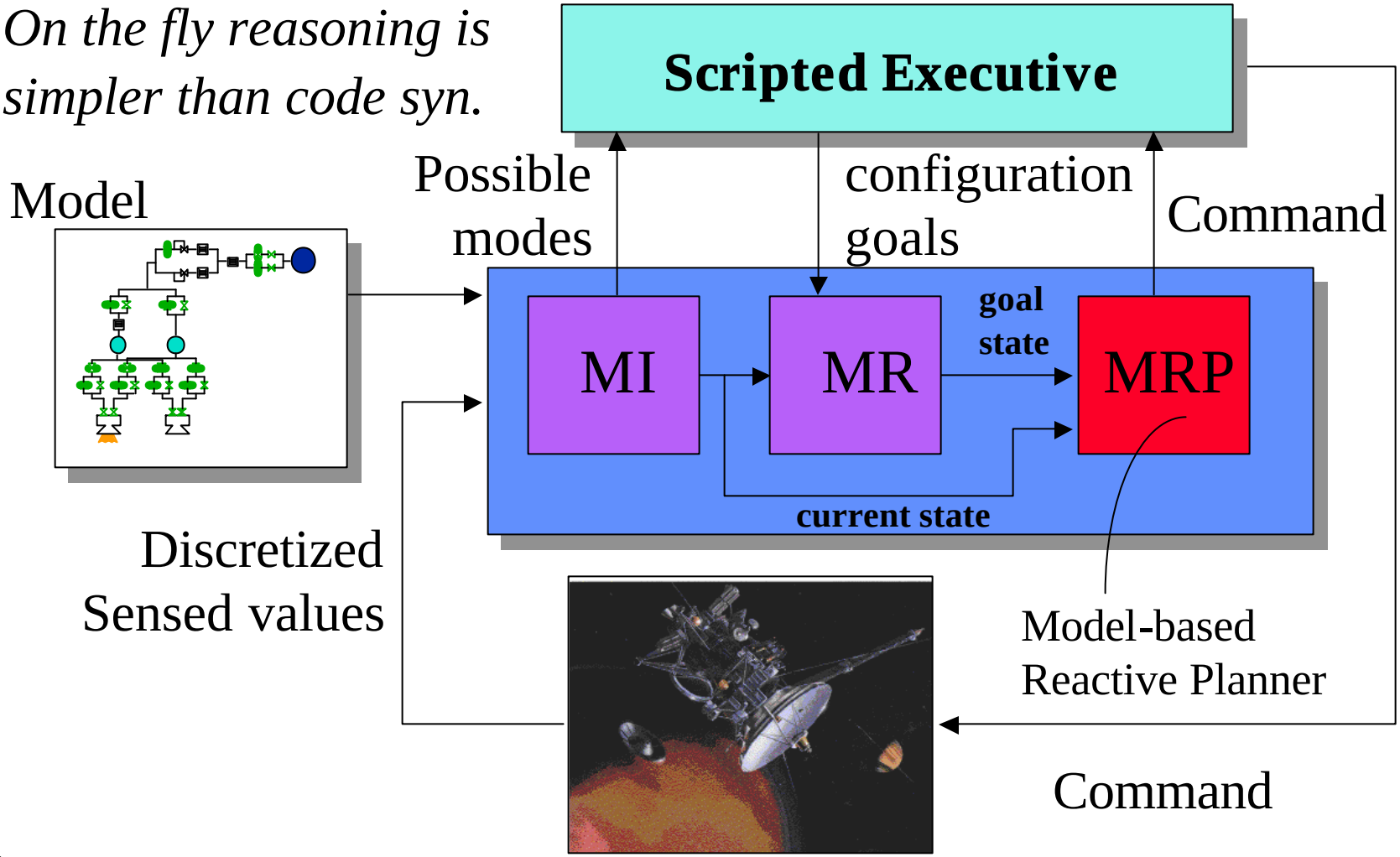
Reconfiguring Modes:

- hardware reconfiguration
- fault recovery
- standby, safing & shutdown
- fault avoidance
- control policy coordination

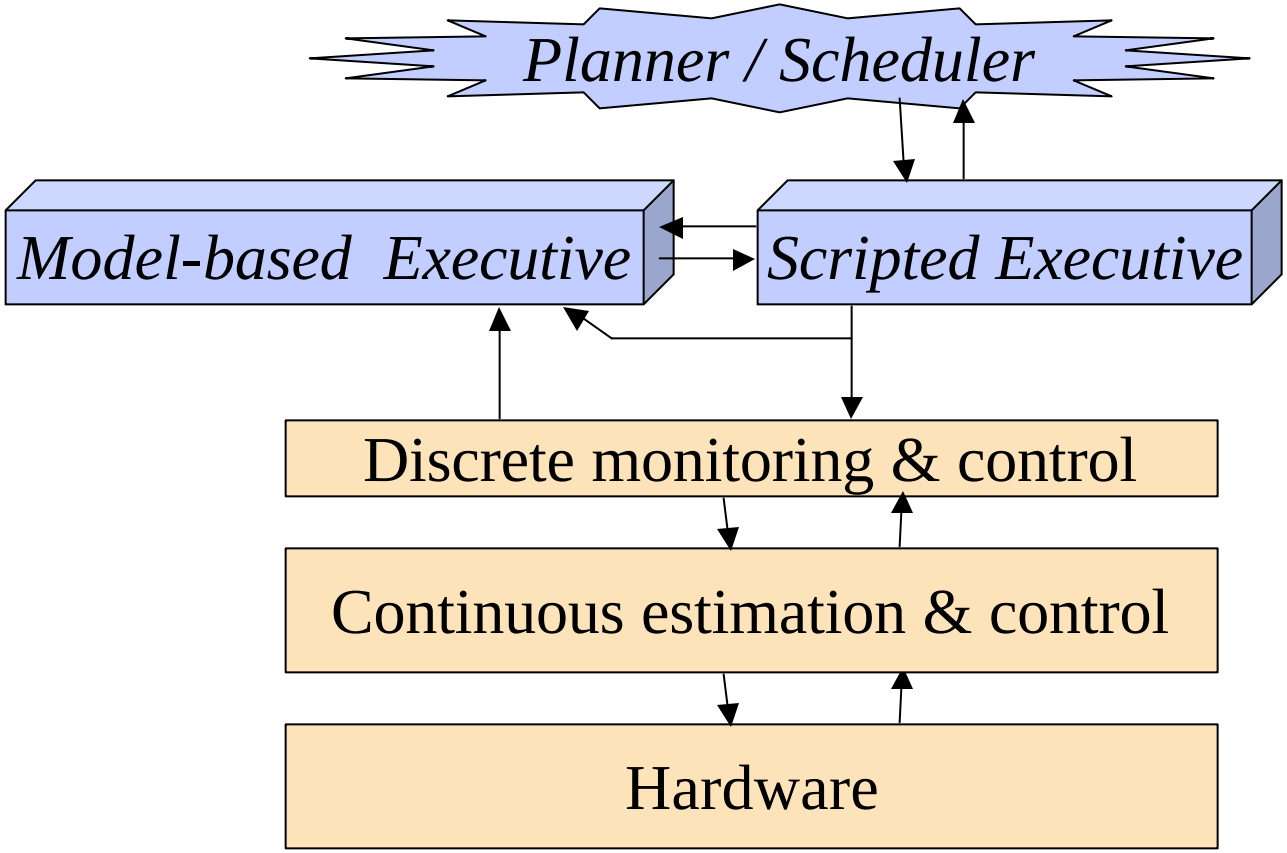


Solution: Part 2
Model-based Deductive Executive

On the fly reasoning is simpler than code syn.



Autonomy architecture with hybrid model-based & scripted executives



Combining Reactivity and Deduction: Can it be achieved?

- **Intractability theorem for planning:** The problem of determining whether a proposition is necessarily true in a nonlinear plan whose action representation is sufficiently strong to represent conditional actions ... is NP Hard.
- "Hoping for the best amounts to arguing that for the particular cases that come up in practice, extensions to current planning techniques will happen to be efficient. My intuition is that this is not the case." -- [Chapman 87]
- **Proposed Solution:** Compile inference into reactive networks, subsumption architectures, ...

But is this solution sufficient?
But is this intuition correct?

Can autonomous systems be built with reactive networks that are low cost & robust?

- Most (semi) autonomous systems operate for long periods of time, often in harsh environments.
  - Saturn probes, automobiles, chemical plants, Antarctic habitats, assembly lines, power networks, computer networks, jetliners, submarines, ...
- Anomalies stackup over time, growing the space of likely anomalous states combinatorially.
  - 1 failure per year over a 7 year mission = n^7 likely anomalies.
- Reactive networks that precompile contingencies to likely anomalies become enormous!

→ **Onboard deduction is required**

However, most NP hard problems aren't hard.

- Satisfiability problems exhibit a phase transition from easily solvable to unsolvable. [Cheeseman et. al. 91]

Satisfiability and CSP problems for physical systems are typically easy

Physical systems ...

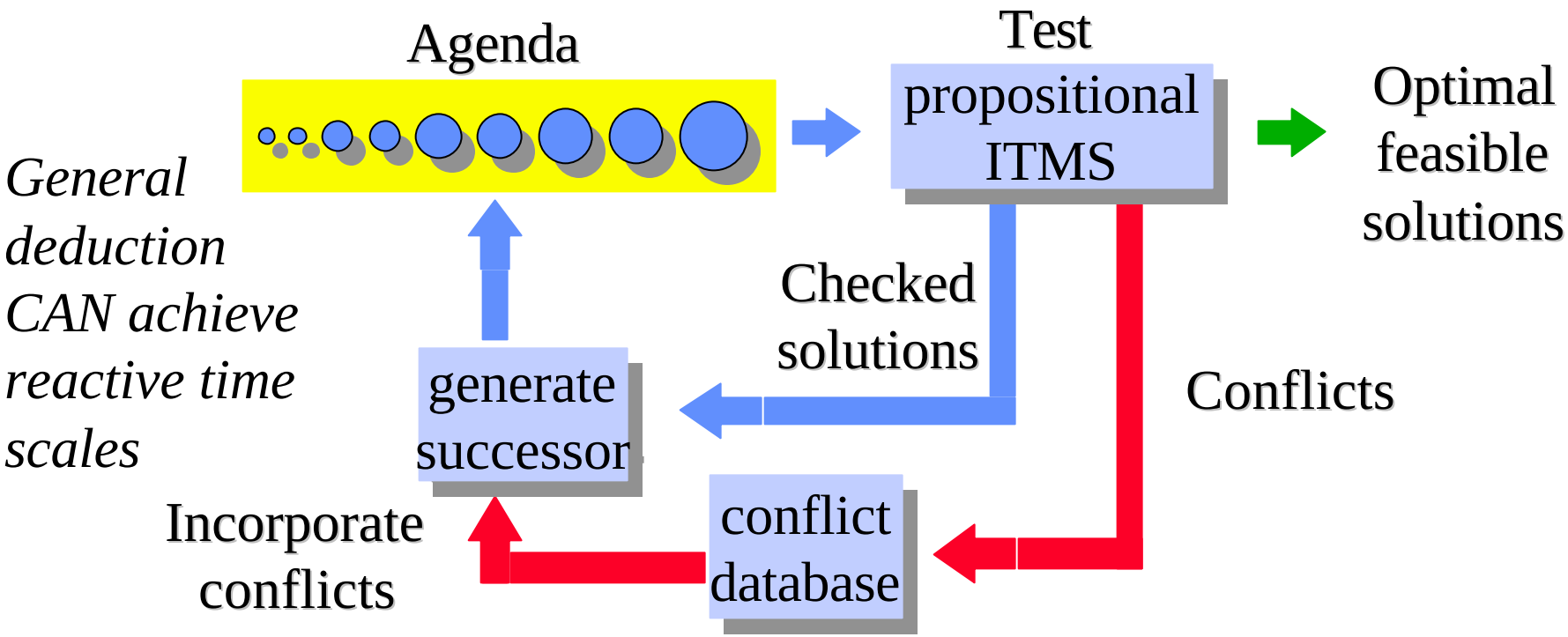
- are causal
- can be modeled with few feedback loops

- Hence, local propagation determines assignments to most state variables -- little search is required to test satisfiability.

Approach: Build RISC-like onboard deductive kernel

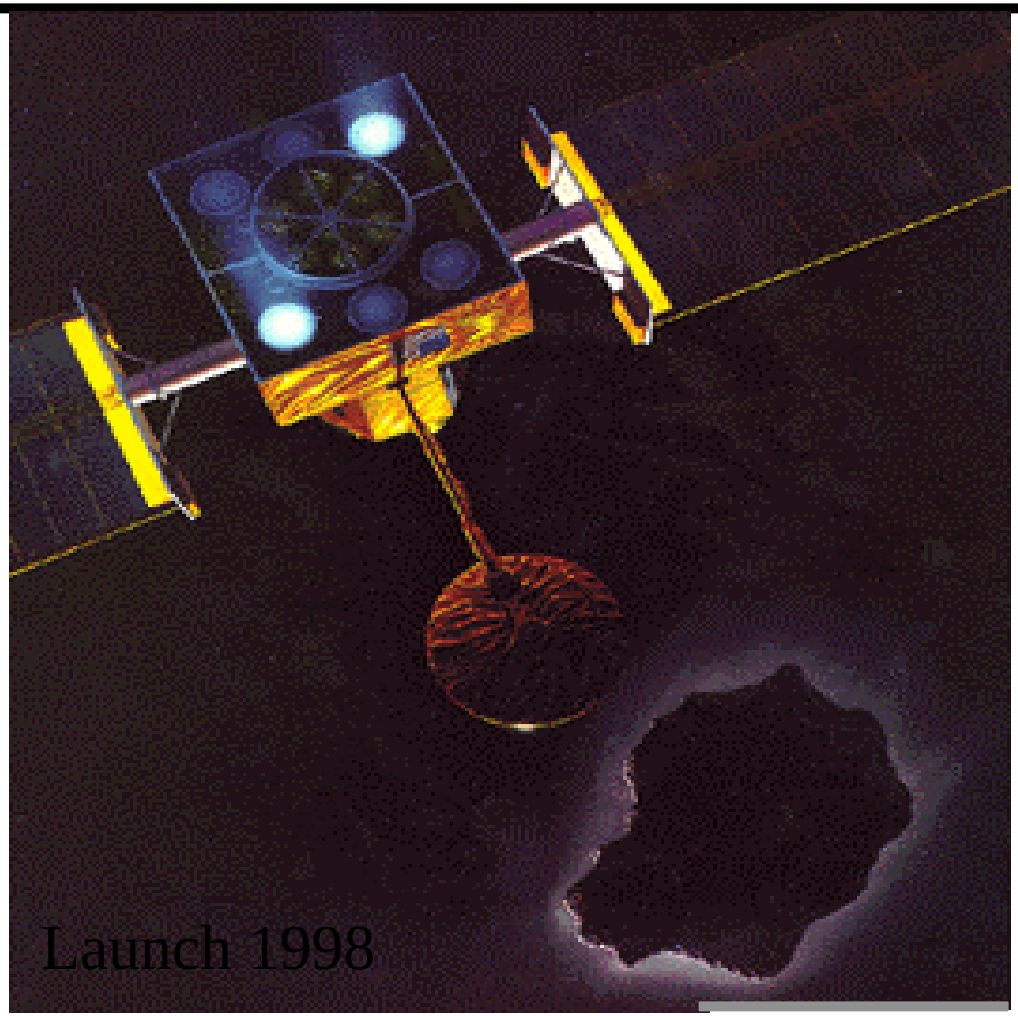
- Propositional SAT or CSP algorithm at its core.
- Minimize SAT queries (e.g., < 10) using highly focused, best first search methods.

Solution: Part 3
Risc-like Best-first, Deductive Kernel



- Tasks and models compiled into propositional logic queries
- Conflicts dramatically focus search
- Careful enumeration grows agenda linearly
- ITMS efficiently tracks state changes in truth assignments

Deep Space One



Summary: Model-based Autonomous Systems

The low cost, rapid prototyping of complex autonomous is enabled by:

- Model-based Programming
- Model-based Deductive, Reactive Execution
  - Self modeling
  - Self configuring
- Model-based Adaptive and Hybrid Systems