No More Band-Aids:
Integrating FM into the Onboard Execution Architecture

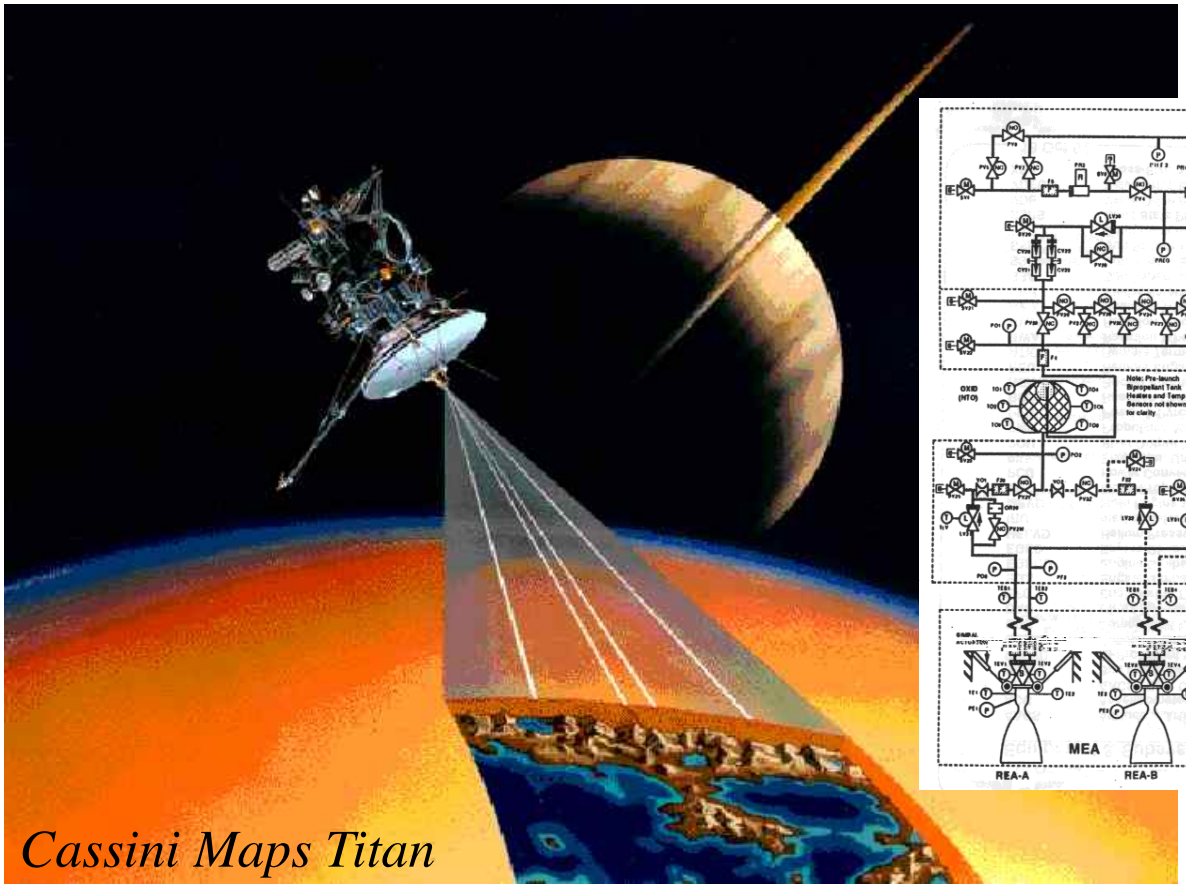
Michel Ingham
Systems Engineering Section
JPL / Caltech

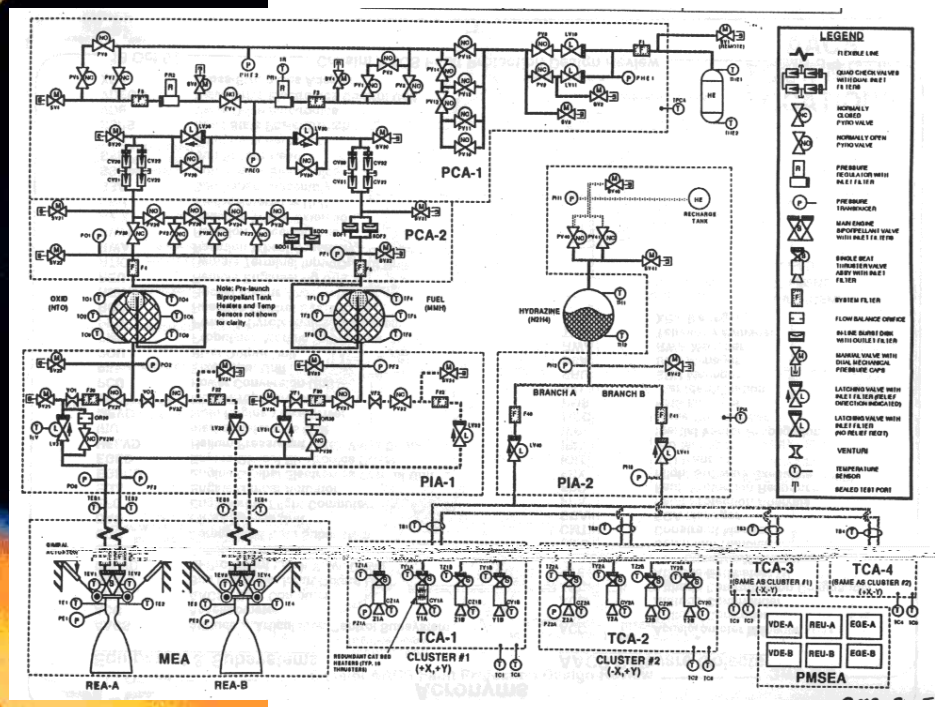
Objectives

- Provide an example of an alternative system architecture, in which Fault Management is “integrated” with nominal execution.

- Encourage thinking outside the usual box, when it comes to Fault Management architecture.

The Complexity Challenge

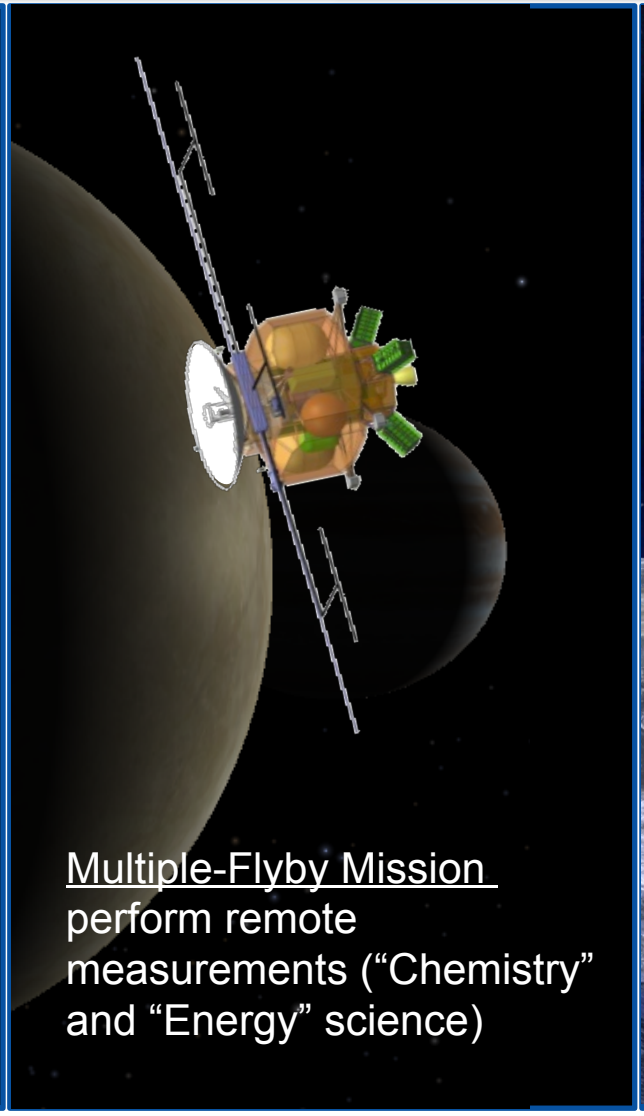


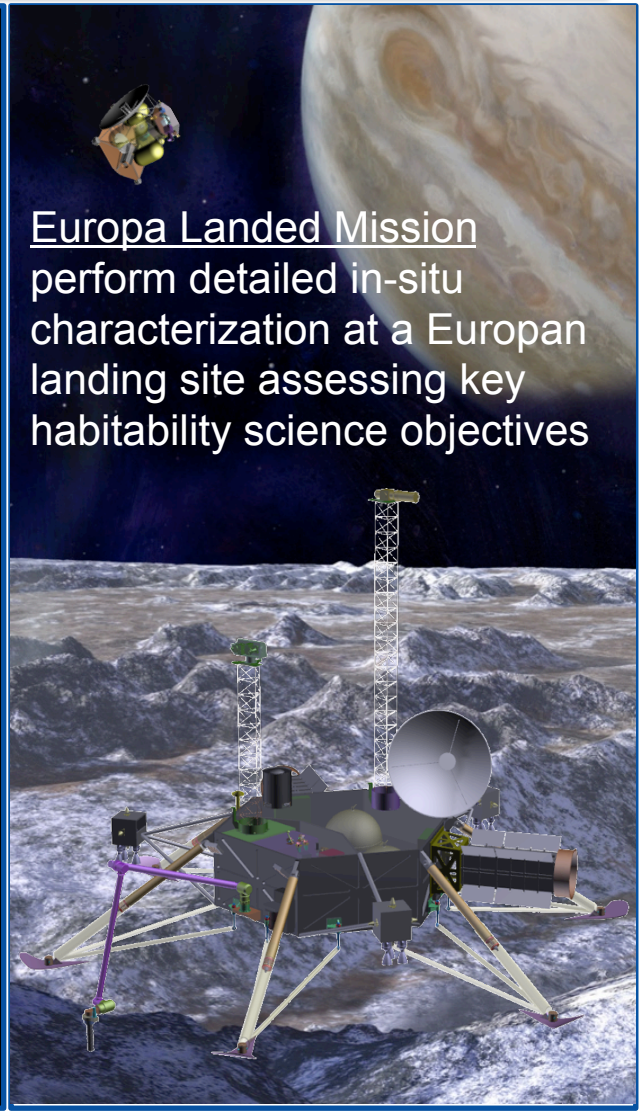


Large collections of devices must work in concert to achieve goals

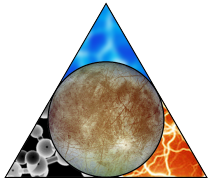
- Devices indirectly observed and controlled.
- Must manage significant redundancy.
- Need quick, robust response to anomalies throughout life.

Europa Mission Concepts



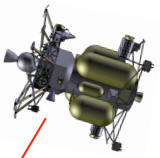


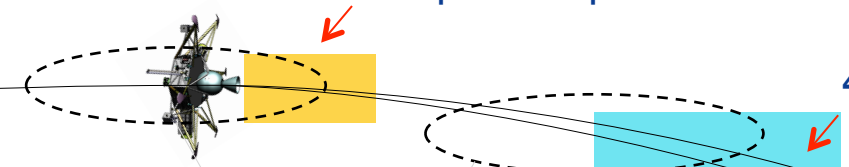
Tiered Landing Site Risk Mitigation



3) **Deorbit Burn**, perform via SRM and monoprop, integrated on high precision IRU w/ real time burn profile updates

1& 2) **Reconnaissance Imagery** and **Site Certification Process** select prioritize list of landing sites in a target zone





4a) **Terrain Relative Navigation (TRN)** determines location and reachable sites

4b) **TRN** performs divert to top priority reachable site

= burn

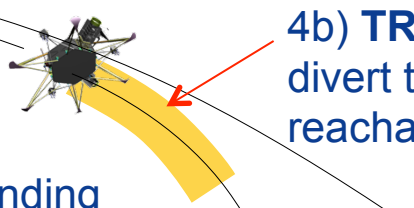
Knowledge Uncertainty

Safe Precision Landing Target Area: 10m x 10m

HA Area: 100m x 100m (before HA divert)

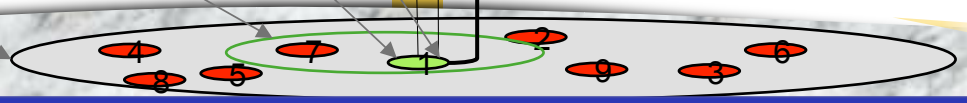
Reachable Sites: 3km x 3km (before TRN divert)

Target Zone: 6km x 3km (before SRM burn)



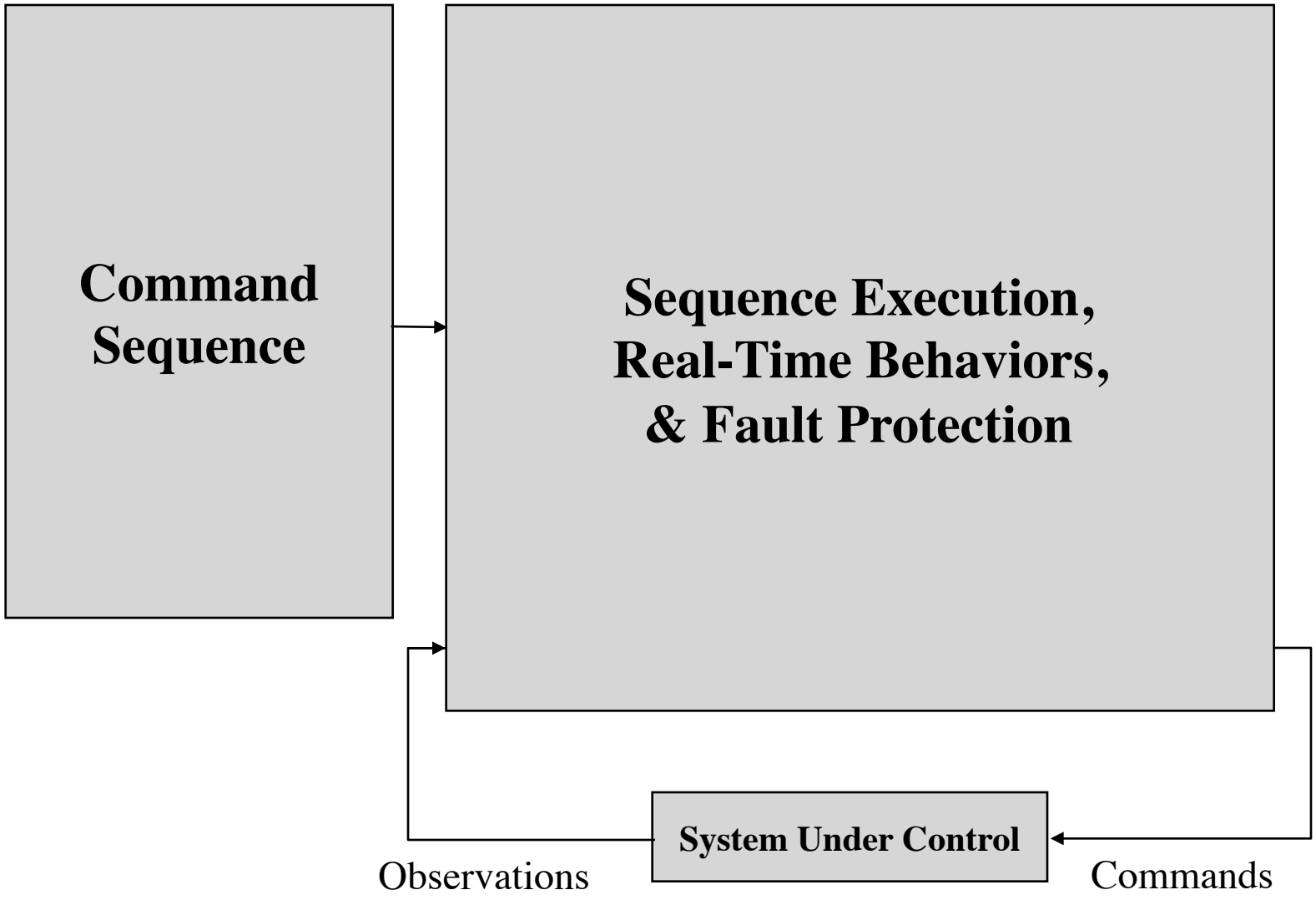
SRM

5) Low altitude **Hazard Avoidance**

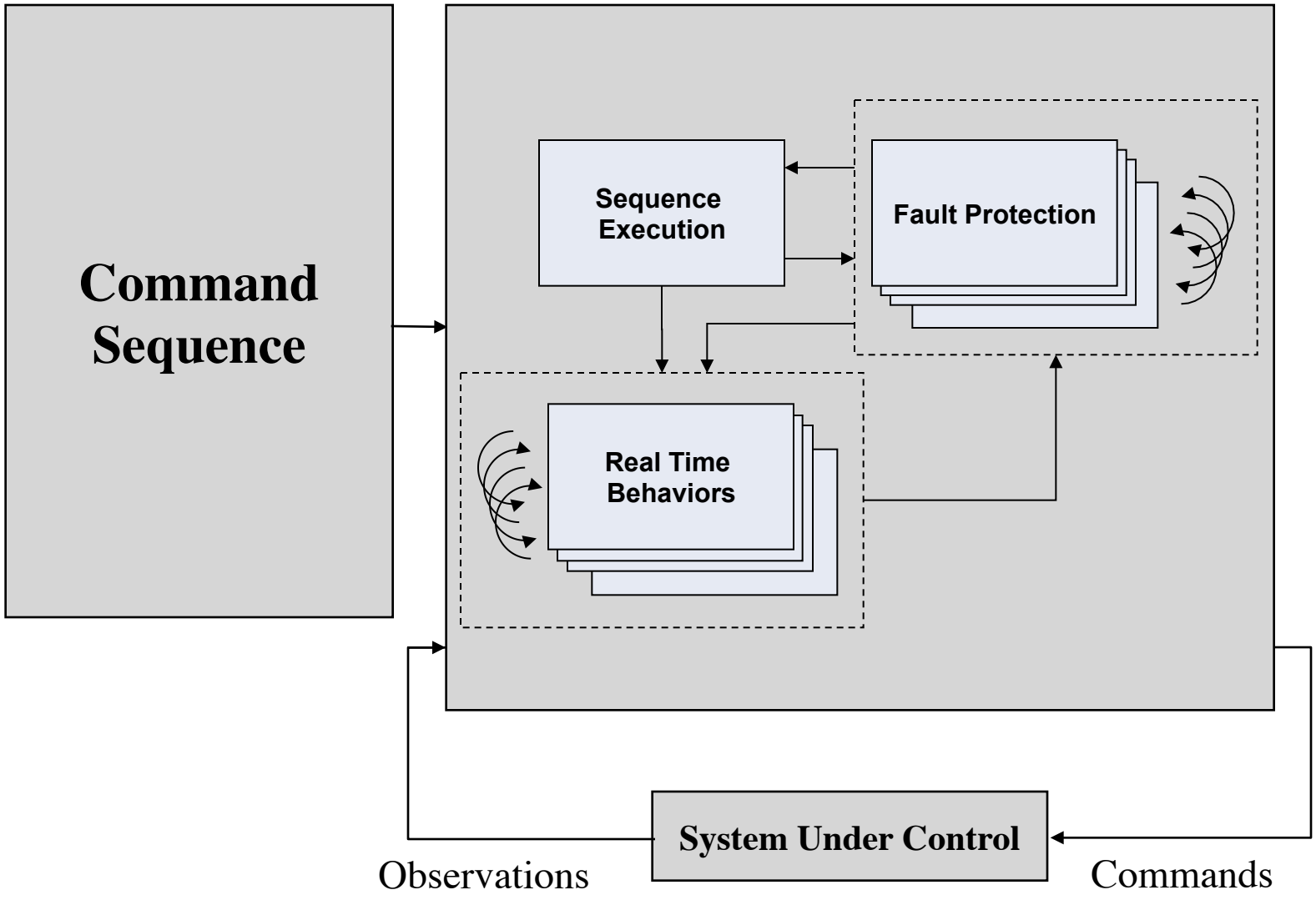


Leverage 15 Years of Technology Development in Human & Mars Programs

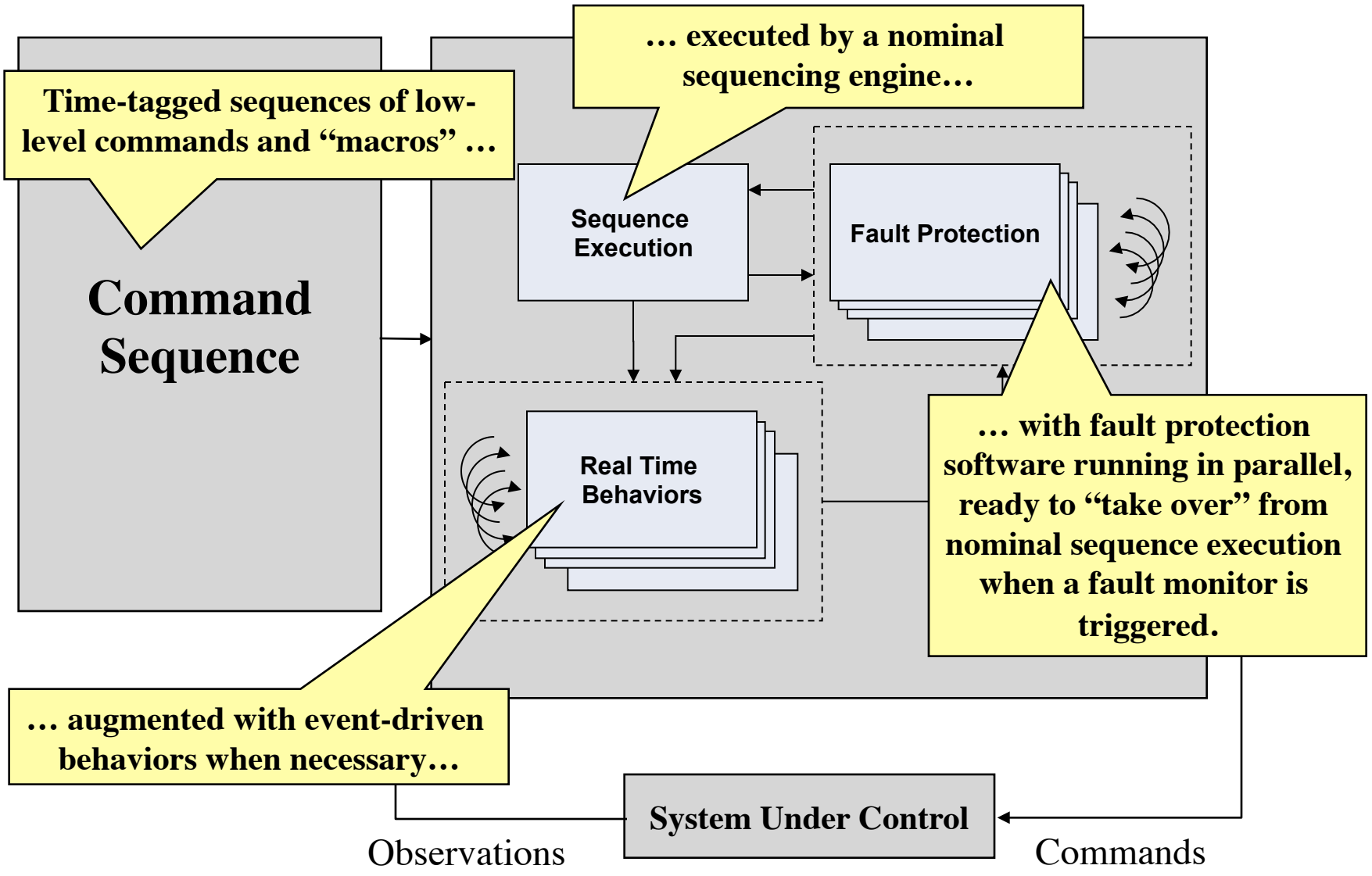
Typical Spacecraft Execution Architecture



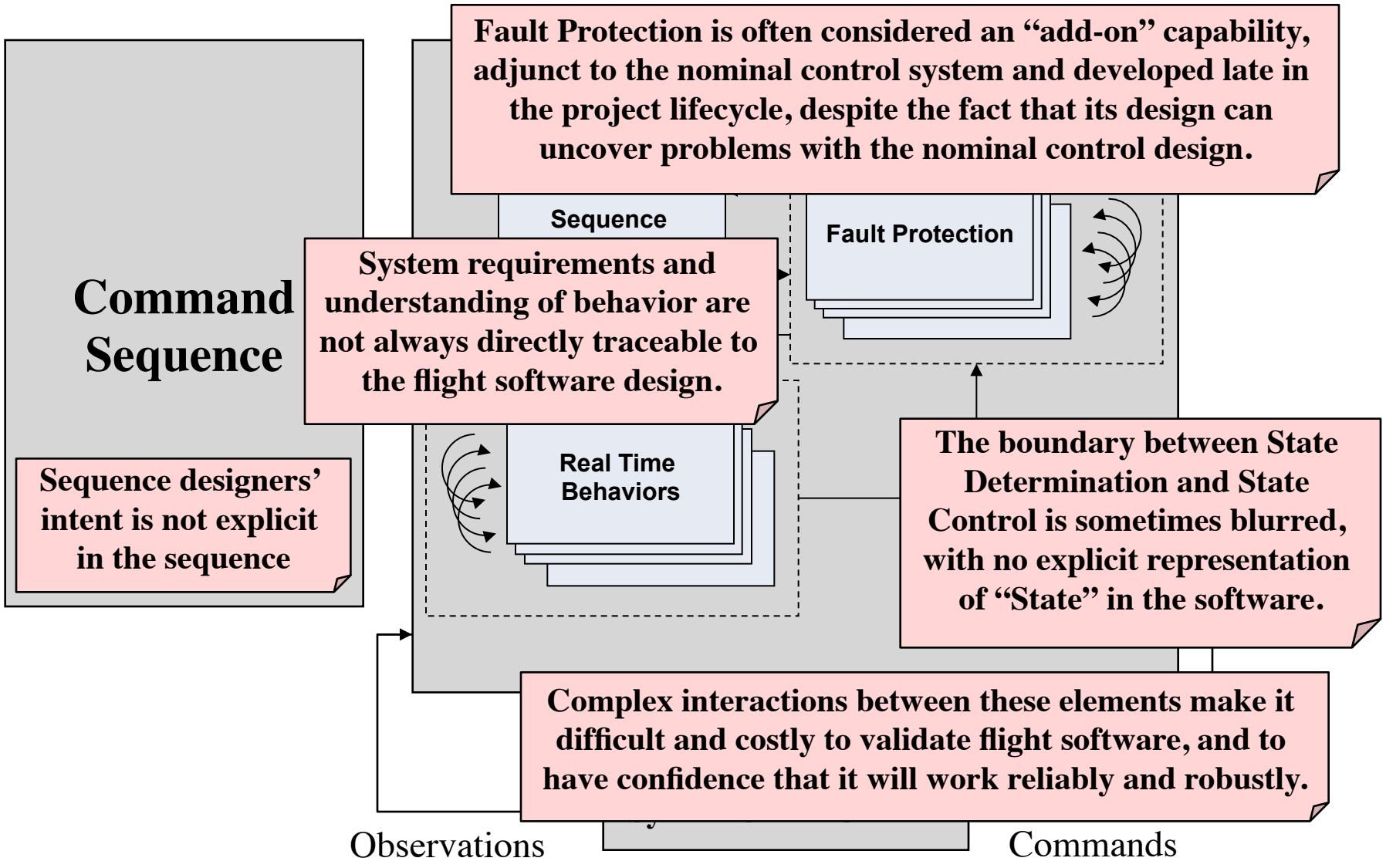
Typical Spacecraft Execution Architecture



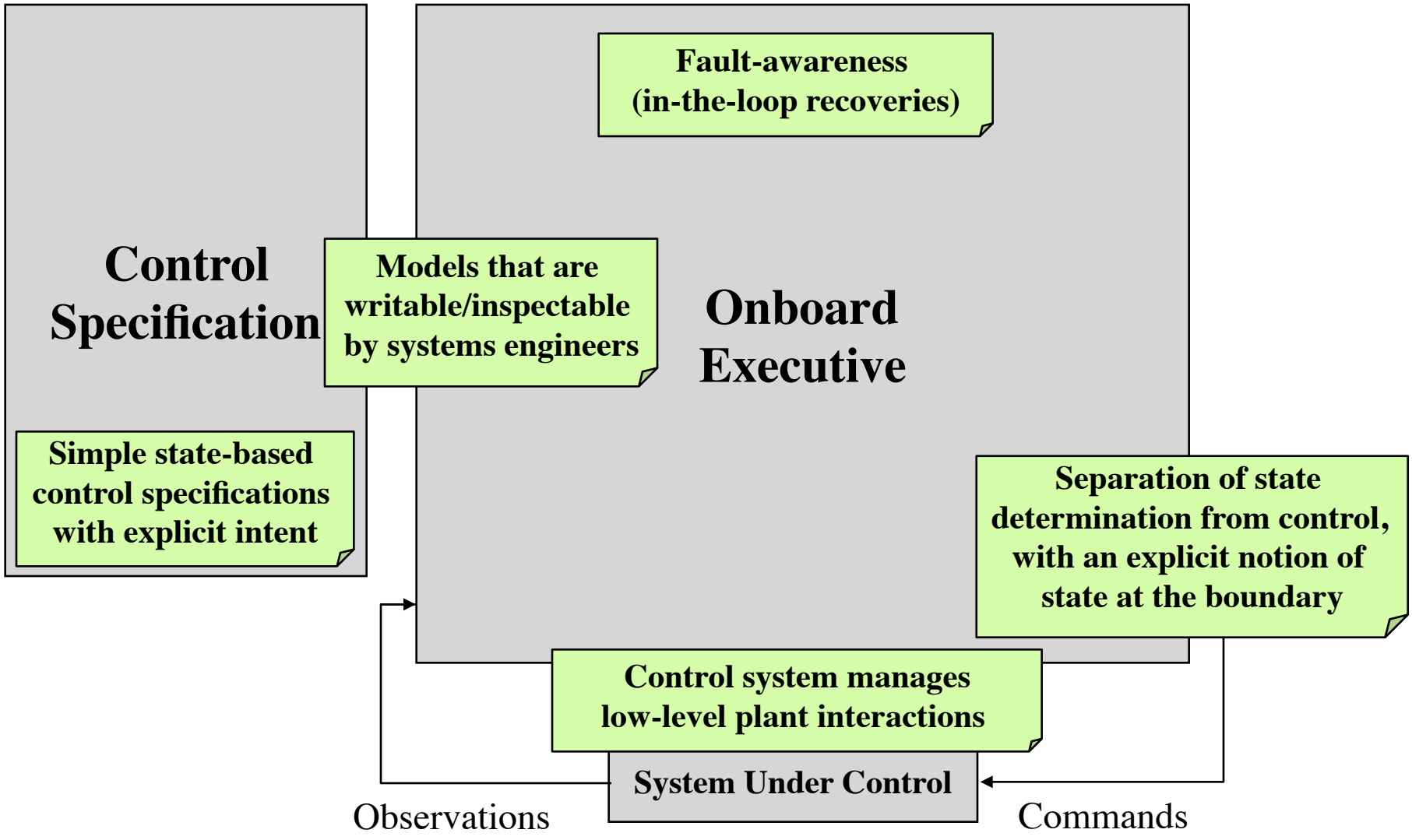
Typical Spacecraft Execution Architecture



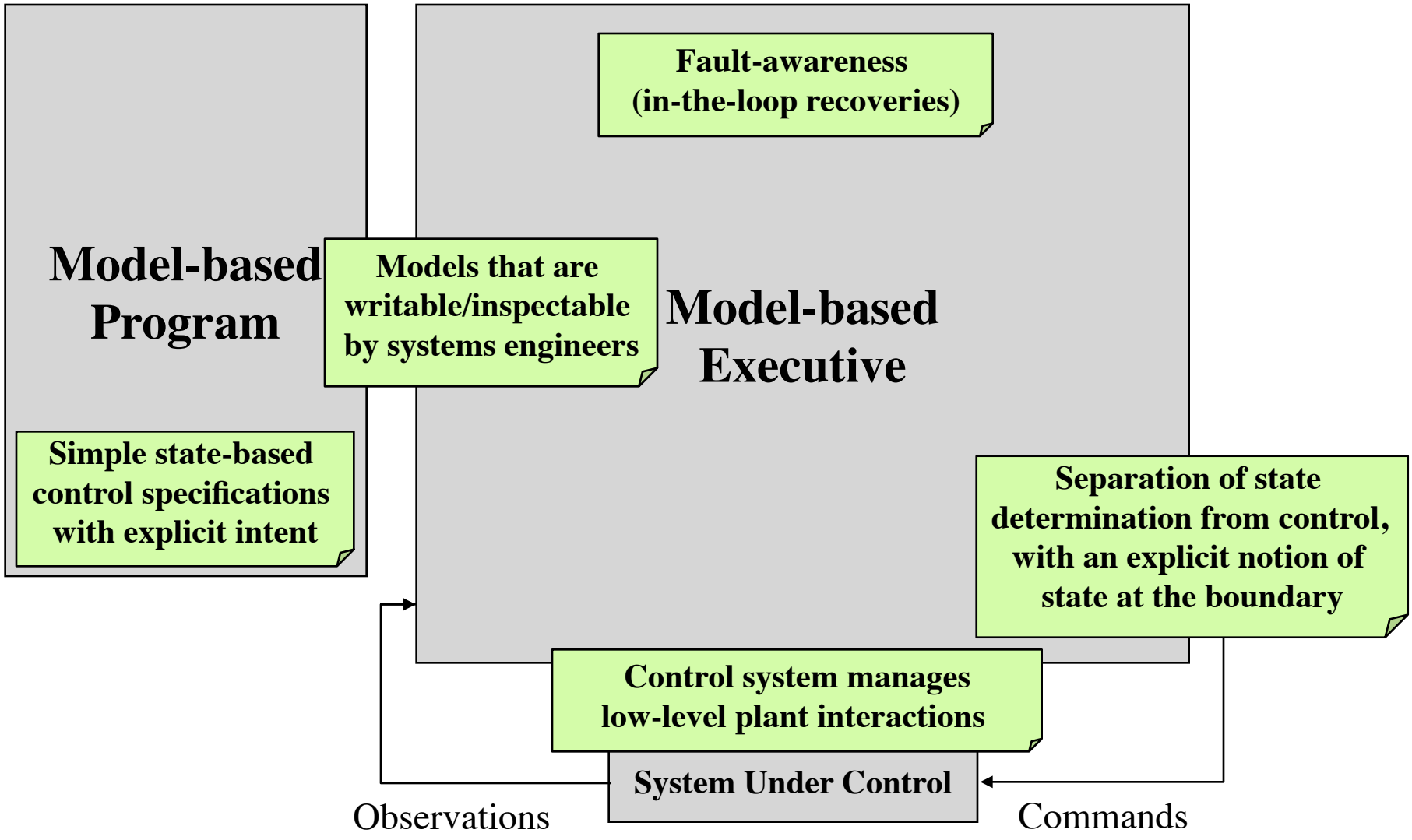
Limitations of the Typical Architecture



Desirable Architectural Features



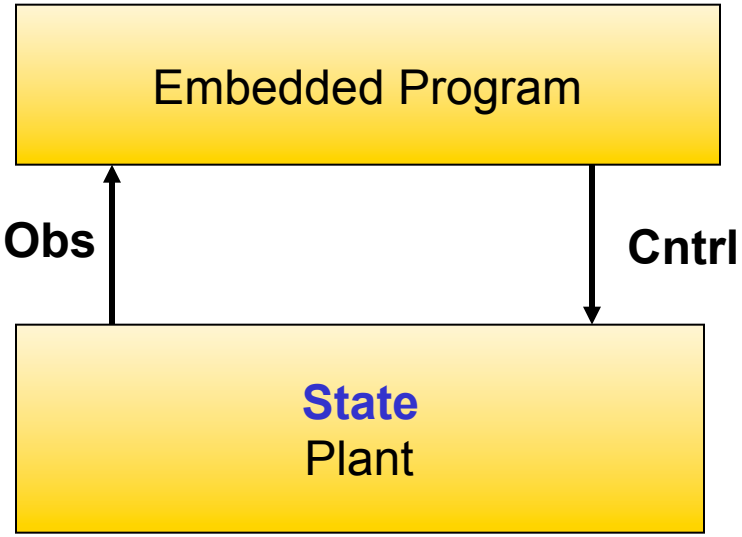
Model-based Programs and Executives Provide These Features



Model-based Programs Reason about State

Embedded programs interact with the system's sensors/actuators:

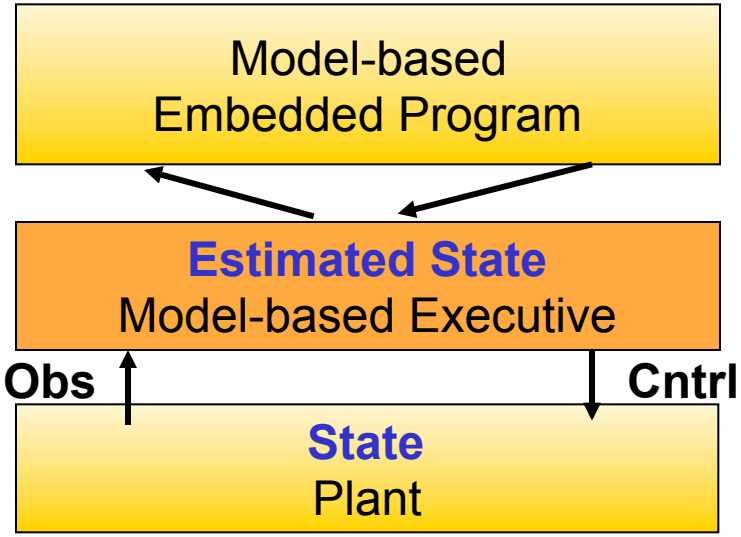
- Read sensors
- Set actuators



Programmers must reason through interactions between state and sensors/actuators.

Model-based programs interact with the system's (hidden) state directly:

- Read state
- Set state

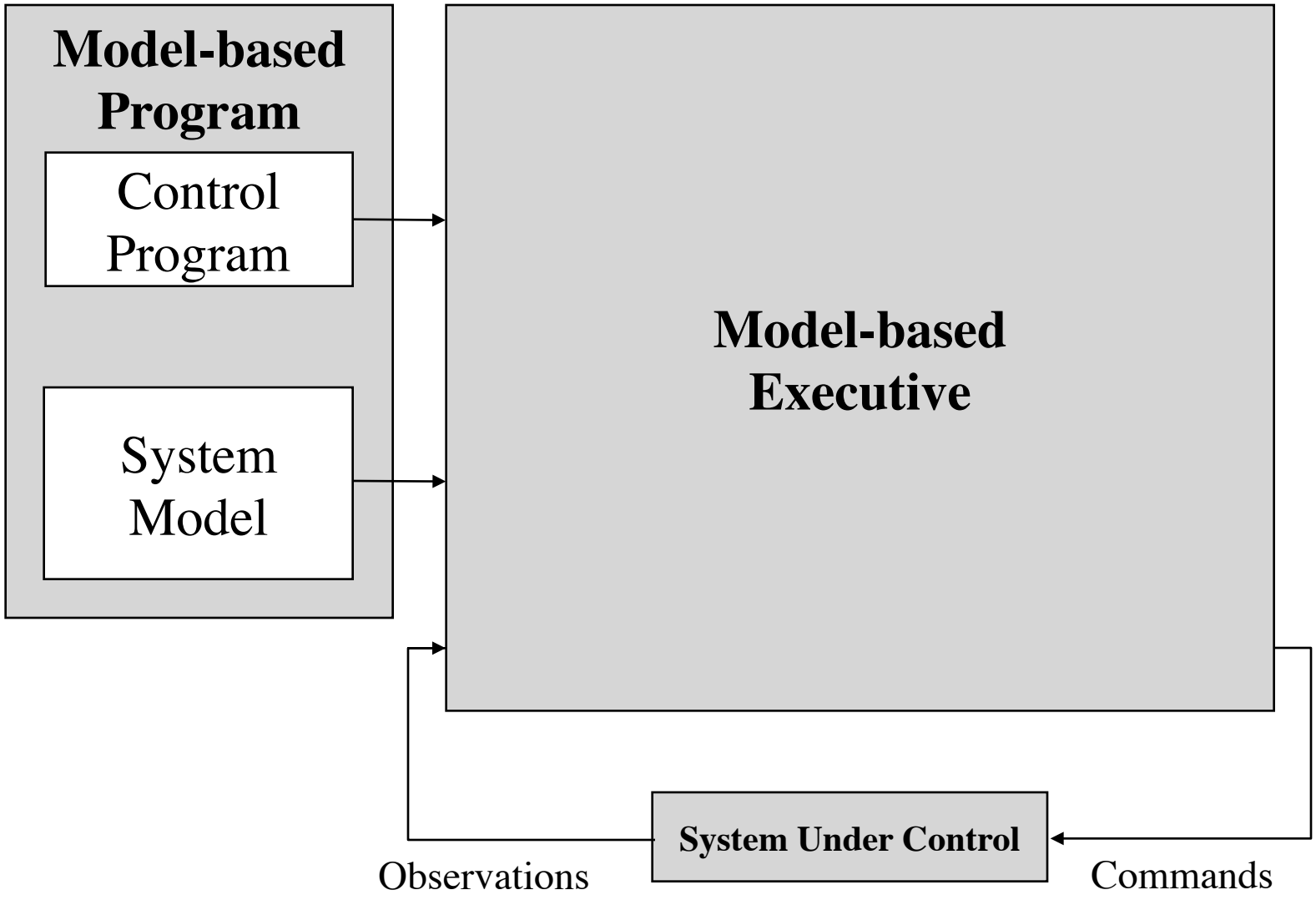


Model-based Executives automatically reason through interactions between states and sensors/actuators.

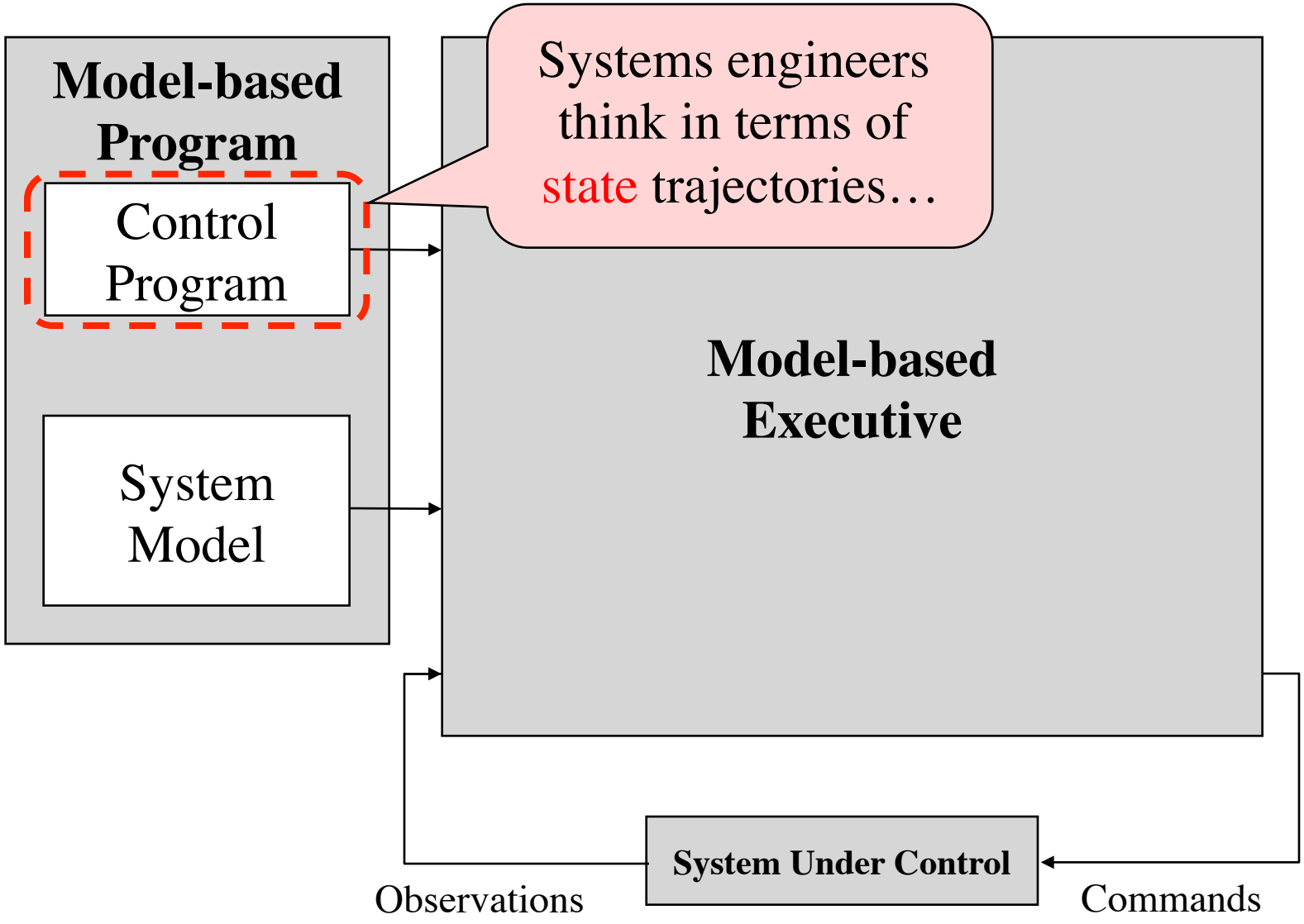
Terminology

- **Model-based Programming** languages elevate the task to storyboarding and modeling.
  - Engineers program their high-level intentions in terms of how they would like the state of the world to evolve.
  - Programmers describe the world (system + environment) using commonsense models of normal and faulty behavior.
- **Model-based Executives** implement these intentions by reasoning on the fly.
  - They continually hypothesize the likely states of the world, given what they observe.
  - They continually plan and execute actions in order to achieve the programmer's intentions.
- **Model-based Autonomy** is the discipline of applying Model-based Programming principles to the control of complex embedded systems.
  - These systems achieve unprecedented robustness ("fault-awareness") by leveraging the capabilities of their Model-based Executives.
  - They automate onboard sequence execution by tightly integrating goal-driven commanding, fault detection, diagnosis and recovery.

Model-based Program



Model-based Program



Control Program: Intent Expressed as Desired State

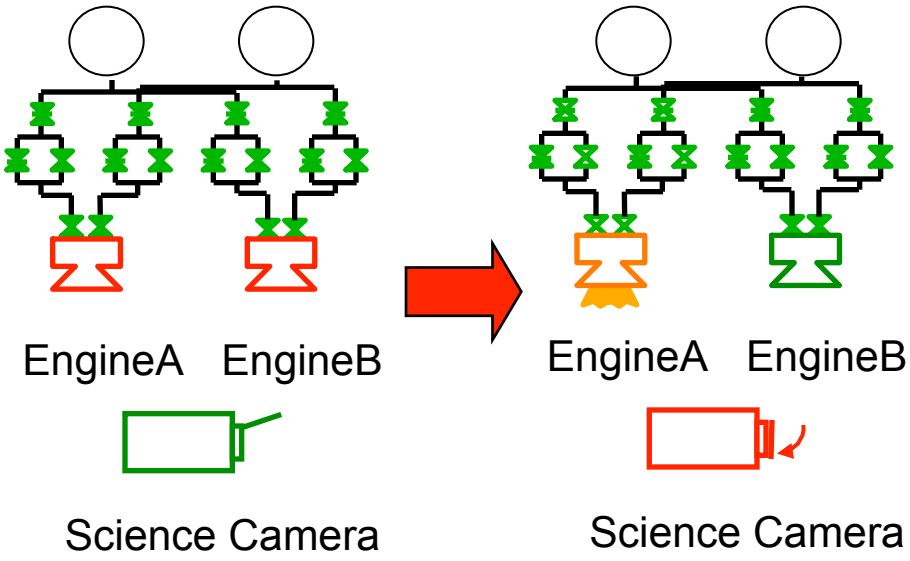
Control Program specifies state trajectories:

– fires one of two engines

– sets both engines to ‘standby’

– prior to firing engine, camera must be turned off to avoid plume contamination

– in case of primary engine failure, fire backup engine instead

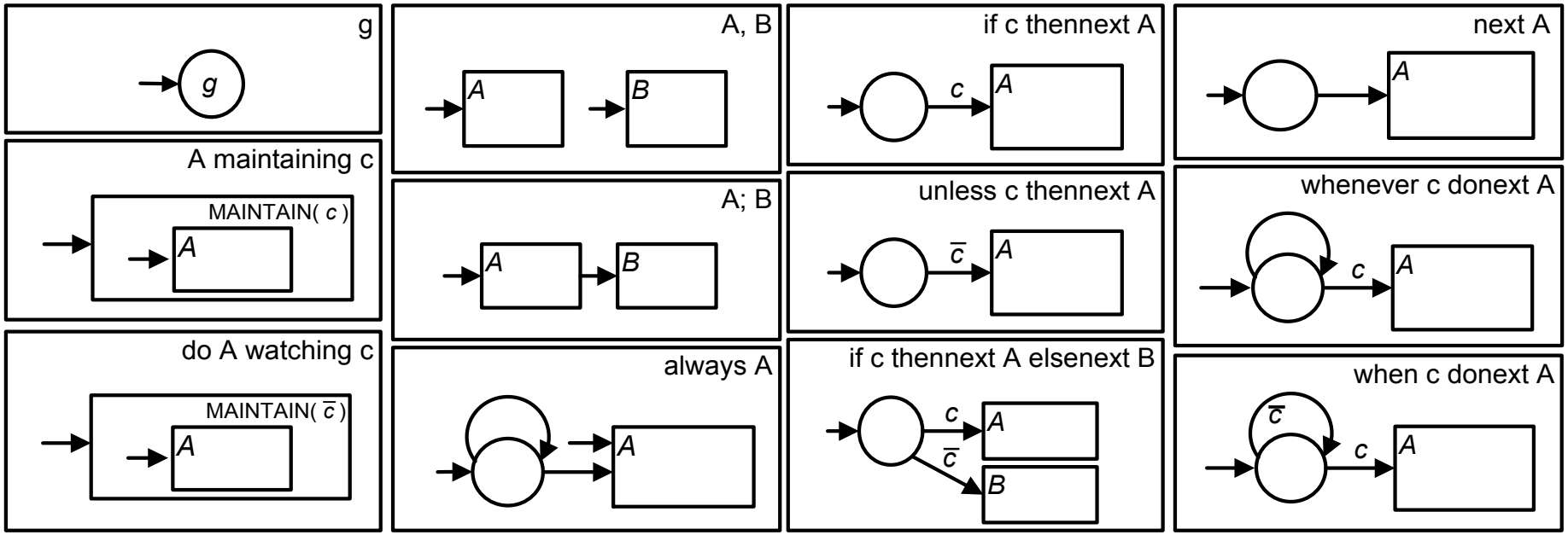



```
OrbitInsert()::
(do-watching ( (EngineA = Firing) OR (EngineB = Firing) )
    (parallel
        (EngineA = Standby)
        (EngineB = Standby)
        (Camera = Off)
        (do-watching (EngineA = Failed)
            (when-donext ( (EngineA = Standby) AND (Camera = Off) )
                (EngineA = Firing) ) )
        (when-donext ( (EngineA = Failed) AND (EngineB = Standby) AND (Camera = Off) )
            (EngineB = Firing) ) ) )
```

RMPL: Reactive Model-based Programming Language

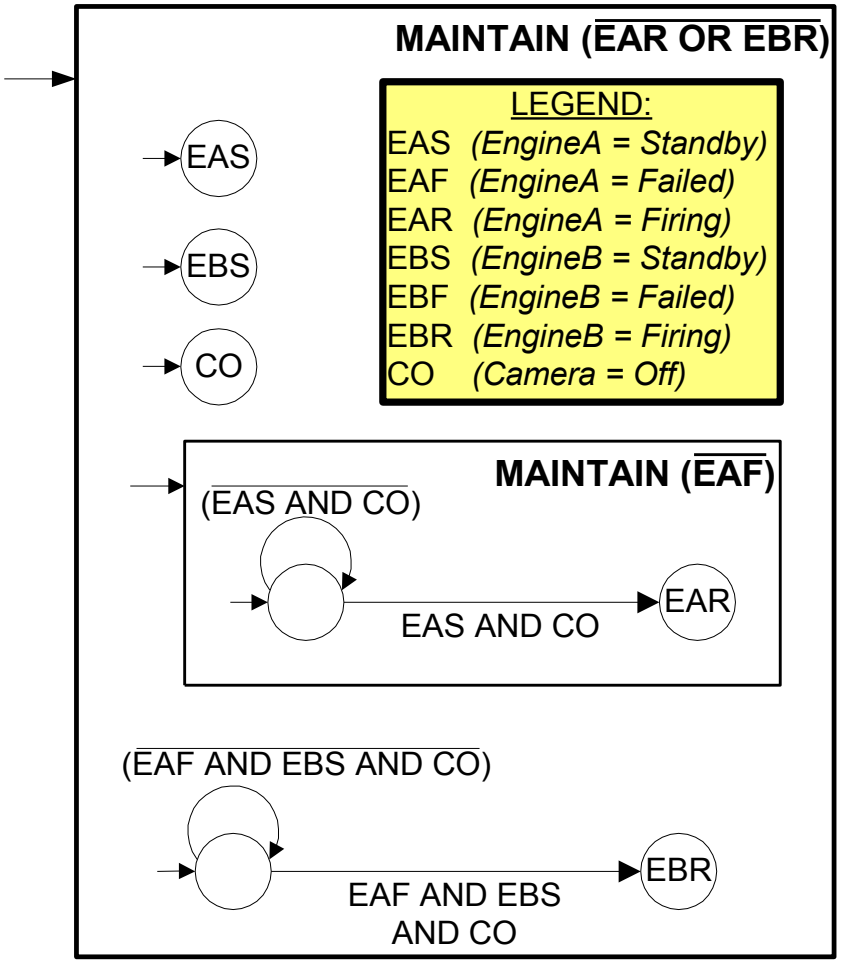
Compiling RMPL to HCA

- Hierarchical Constraint Automata (HCA): graphical specification language for control programs, in the spirit of StateCharts
- Writable, inspectable by systems engineers
- Directly executable by Control Sequencer

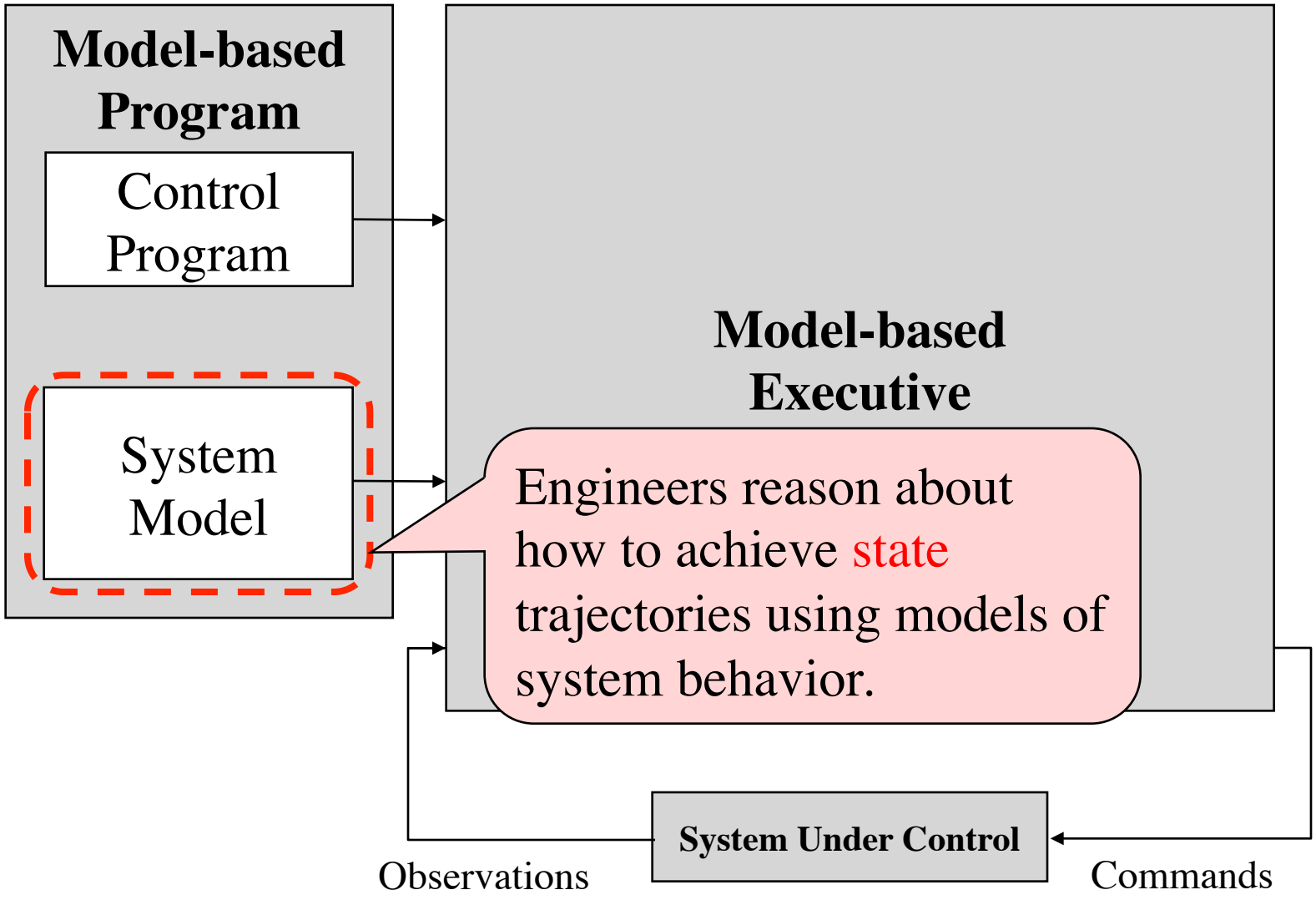


Compiling RMPL to HCA

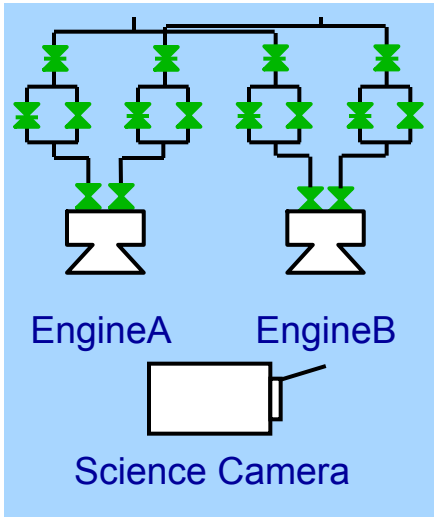
```
OrbitInsert()::
(do-watching ((EngineA = Firing) OR
              (EngineB = Firing))
    (parallel
        (EngineA = Standby)
        (EngineB = Standby)
        (Camera = Off)
        (do-watching (EngineA = Failed)
            (when-donext ( (EngineA = Standby) AND
                           (Camera = Off) )
                (EngineA = Firing)))
        (when-donext ( (EngineA = Failed) AND
                       (EngineB = Standby) AND
                       (Camera = Off) )
            (EngineB = Firing))))
```

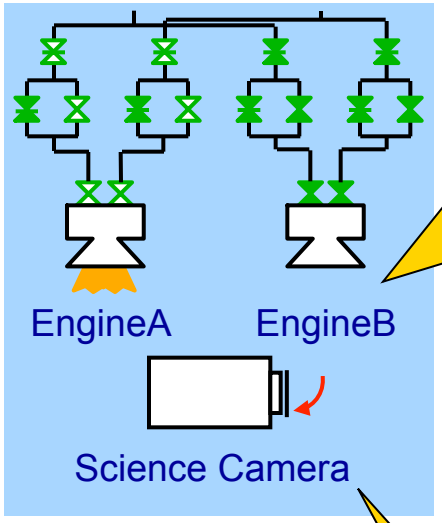


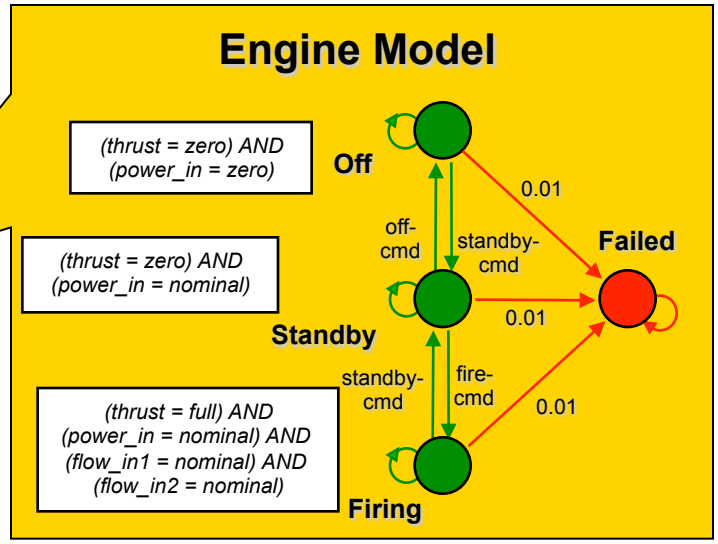
Model-based Program

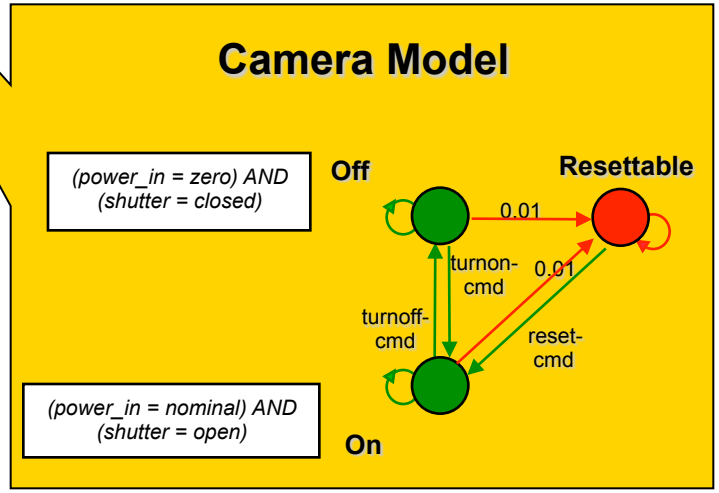


System Model: Formal Descriptions of State Behavior







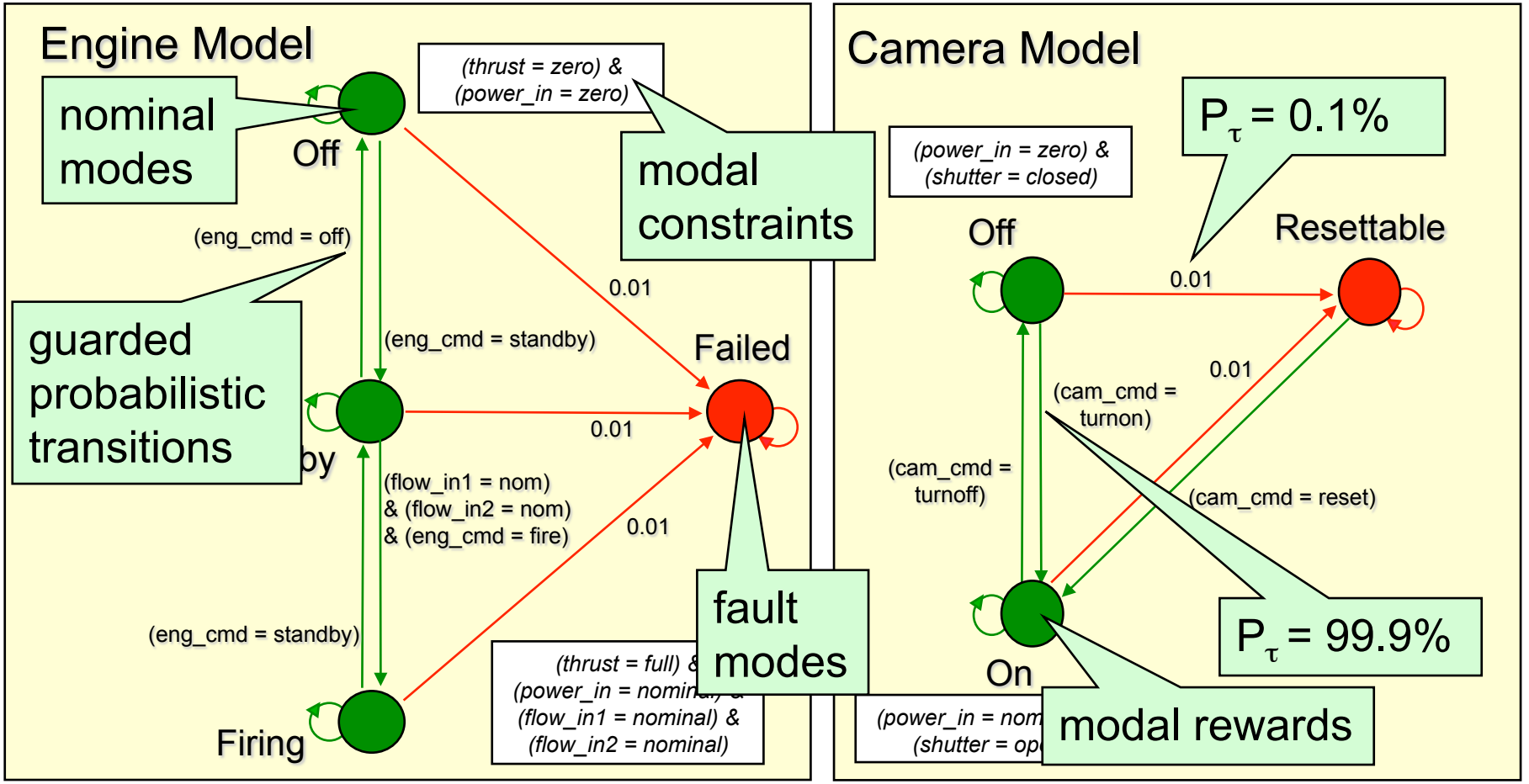


System Model describes behavior of each component:

– nominal and off-nominal behavior
– qualitative constraints
– probabilistic transitions
– costs/rewards

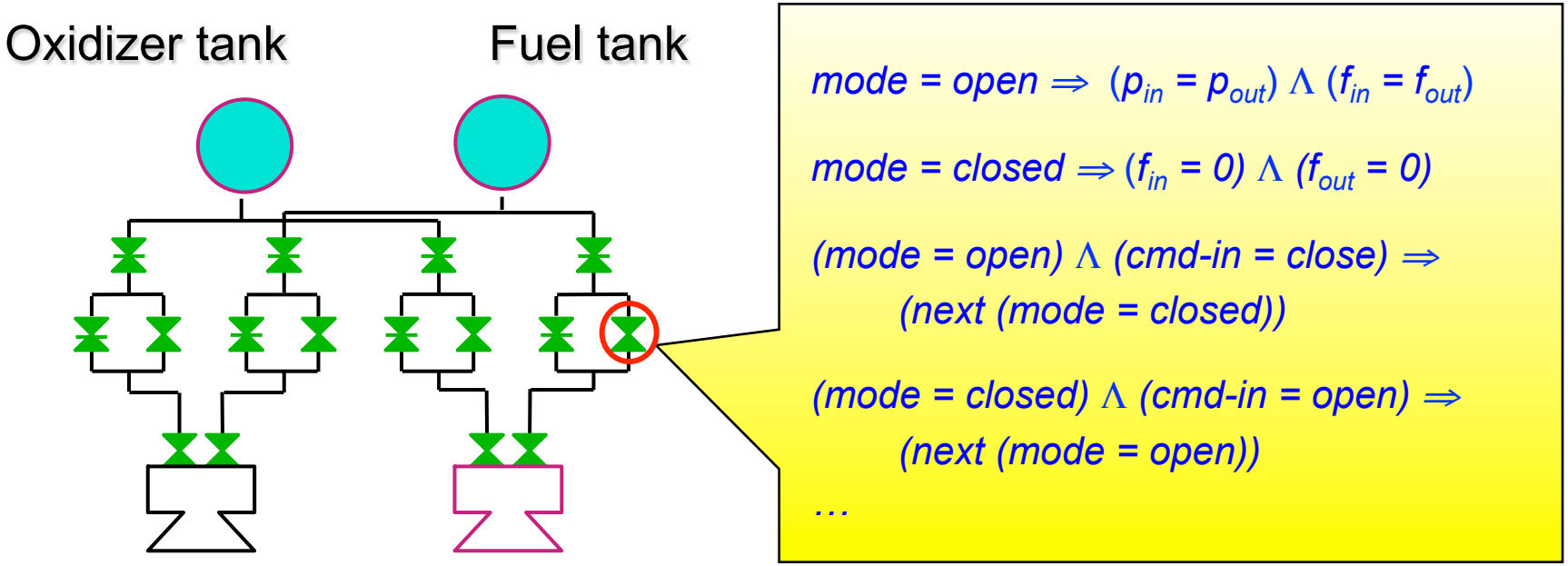
One state machine per component, operating concurrently

Concurrent Constraint Automata

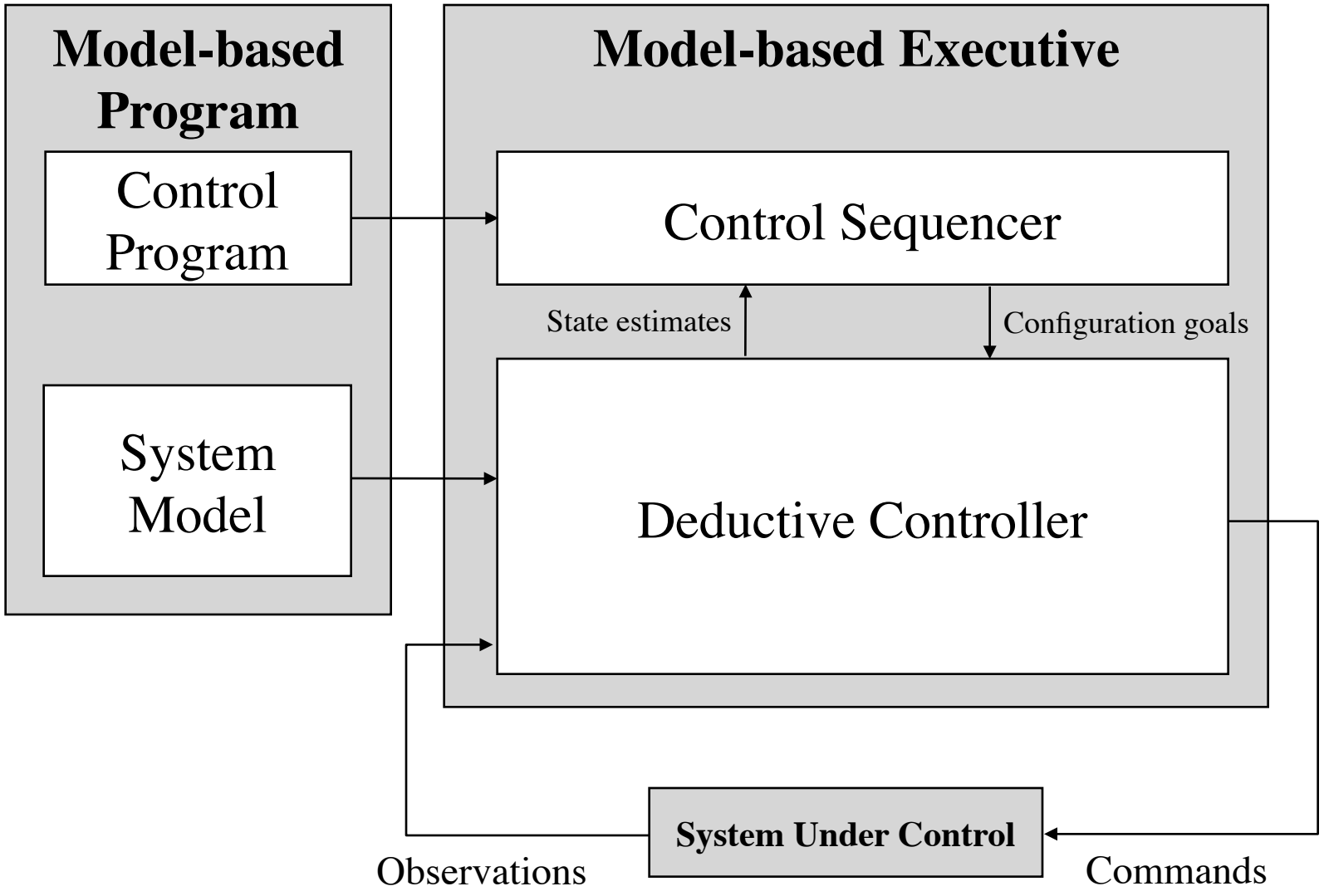


Translating CCA to Propositional Logic

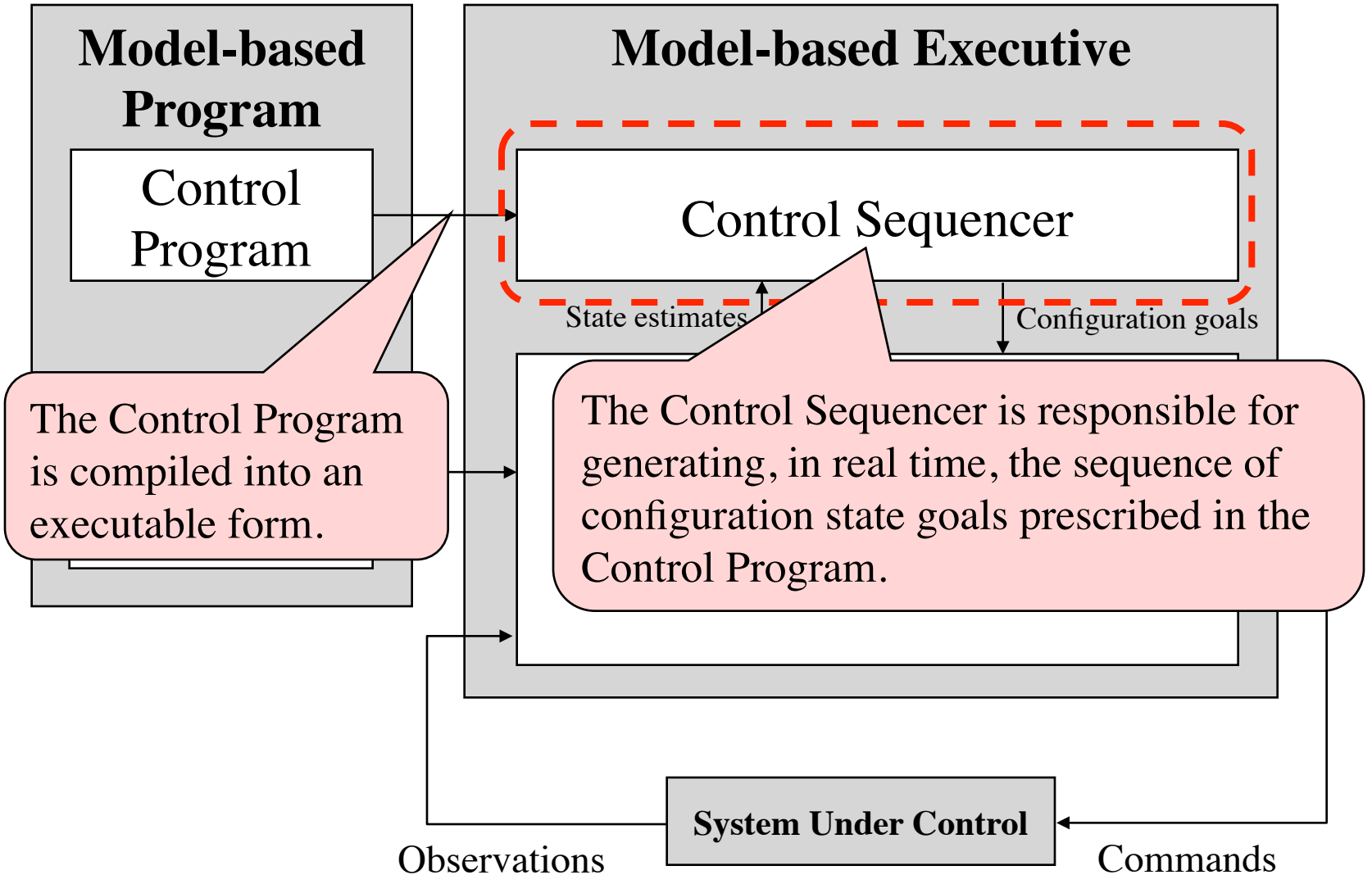
- System Model captured as CCA
- CCA representation translates directly to clauses in propositional logic
- Logical representation is used by reasoning algorithm in Deductive Controller



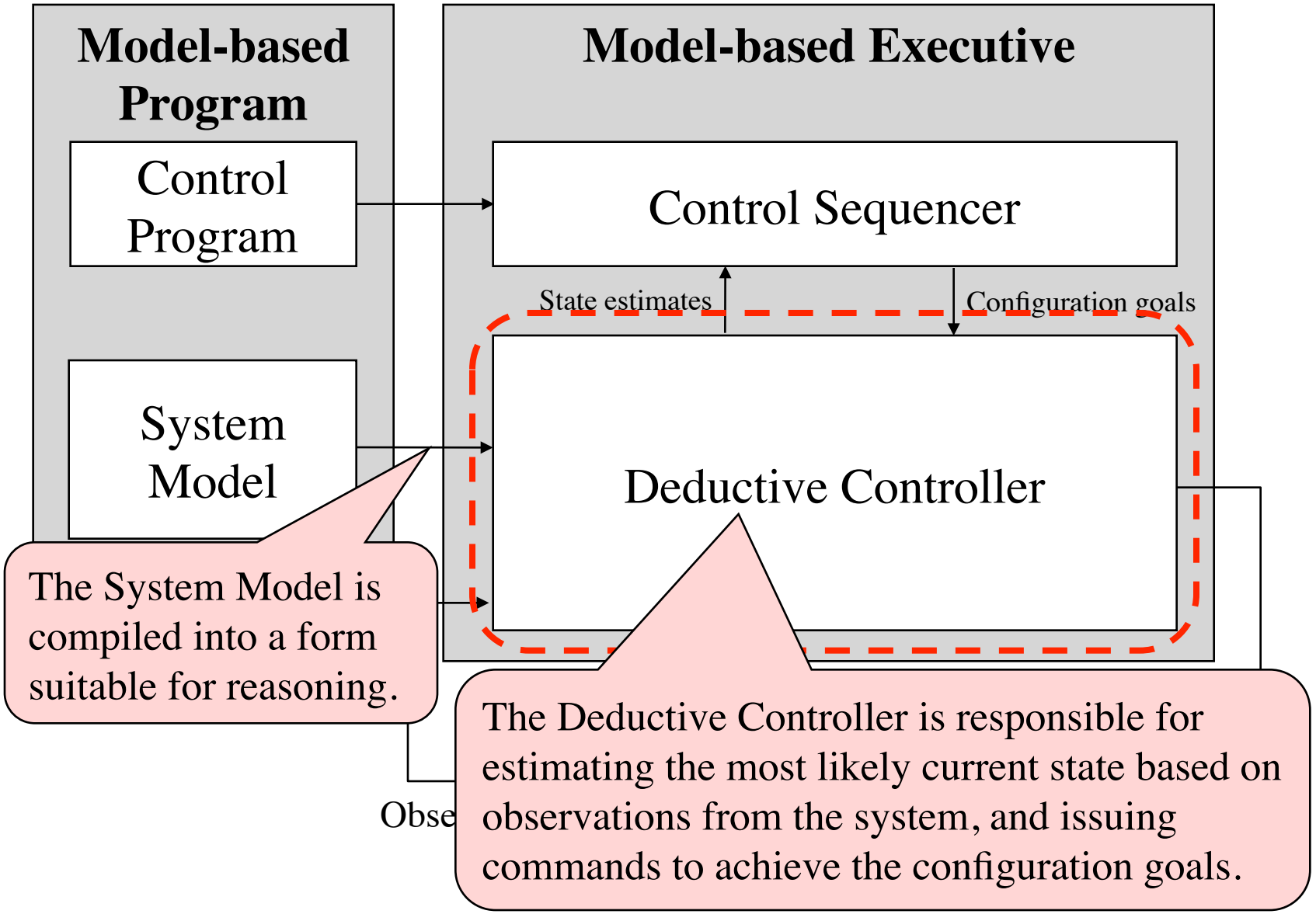
Model-based Executive



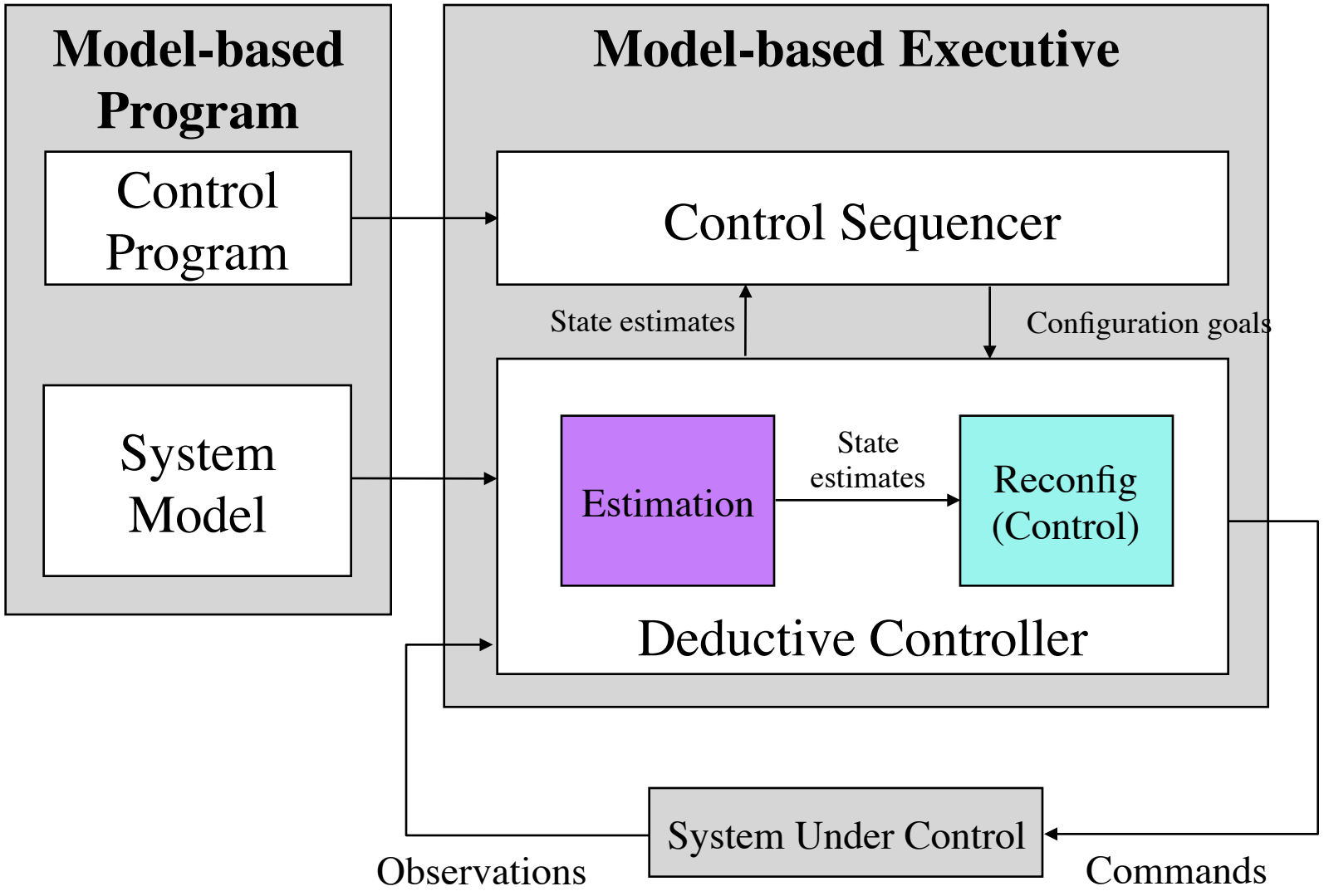
Model-based Executive



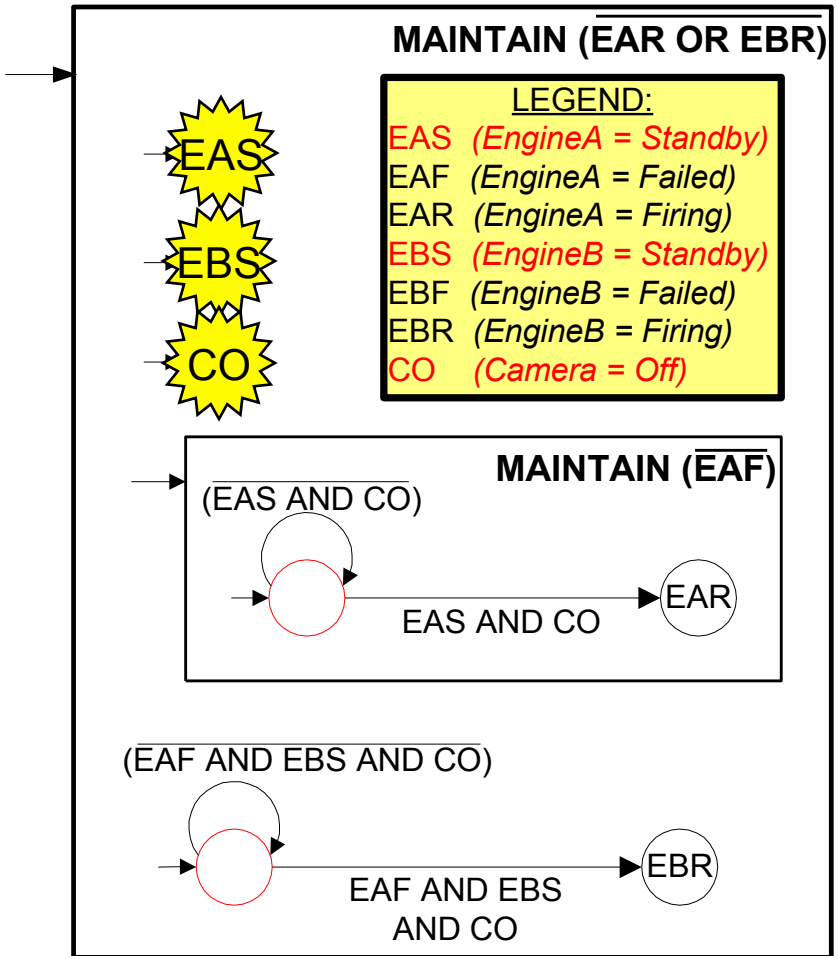
Model-based Executive

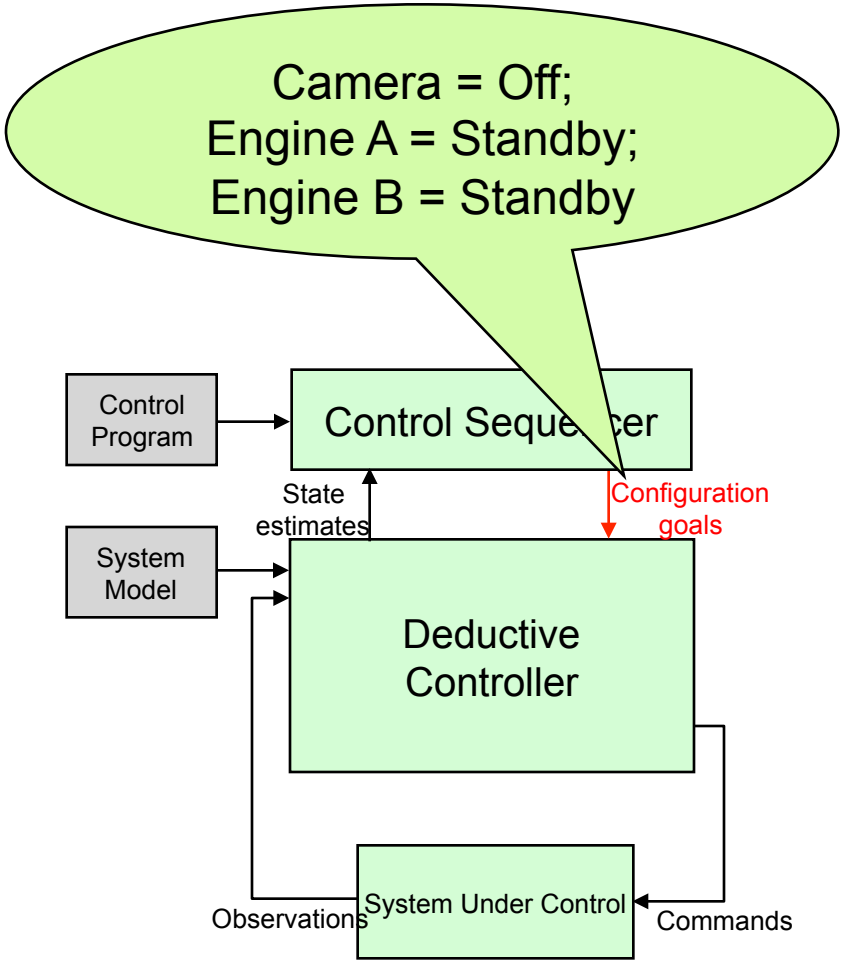


Model-based Executive

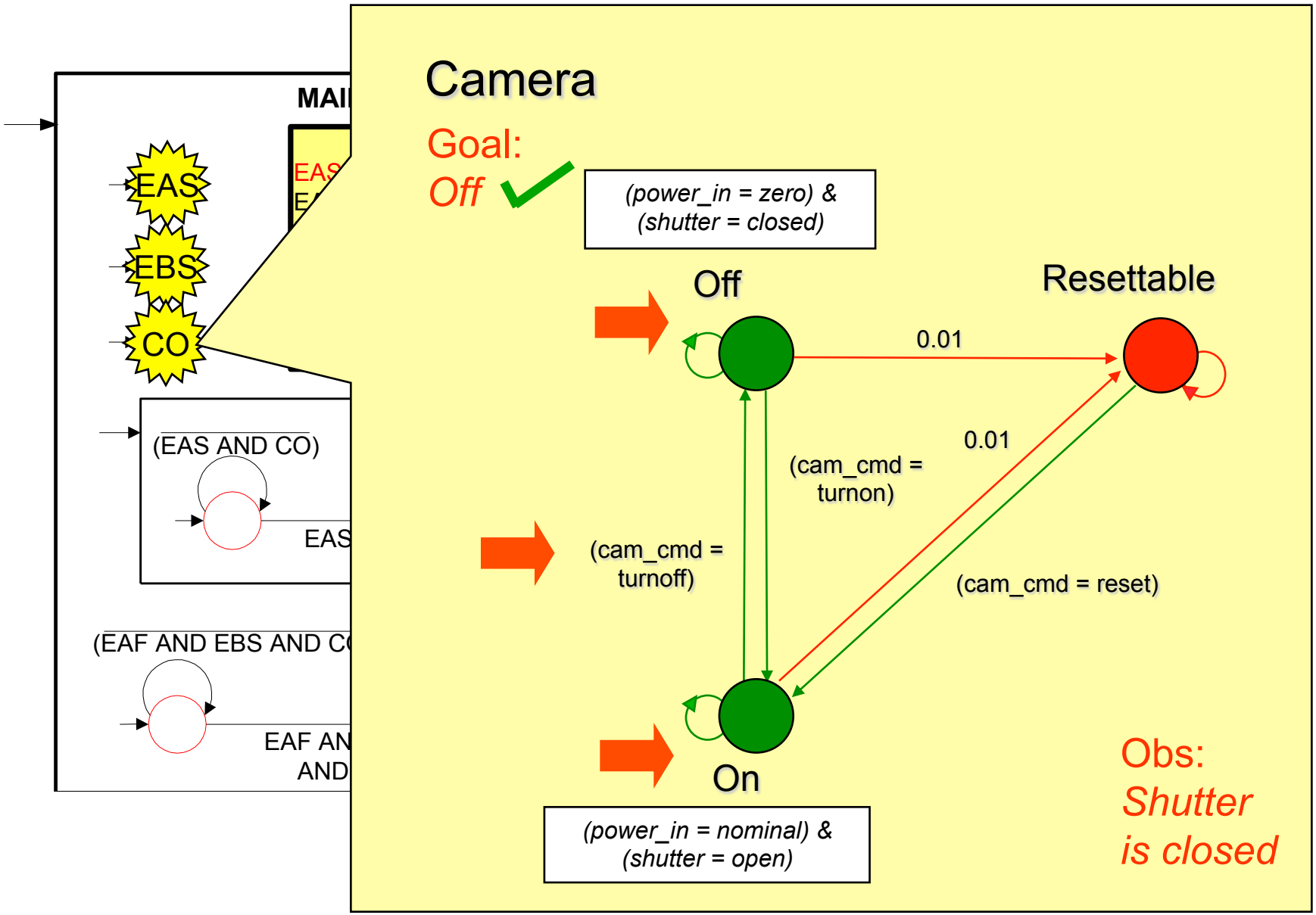


Executing HCA - Step 1

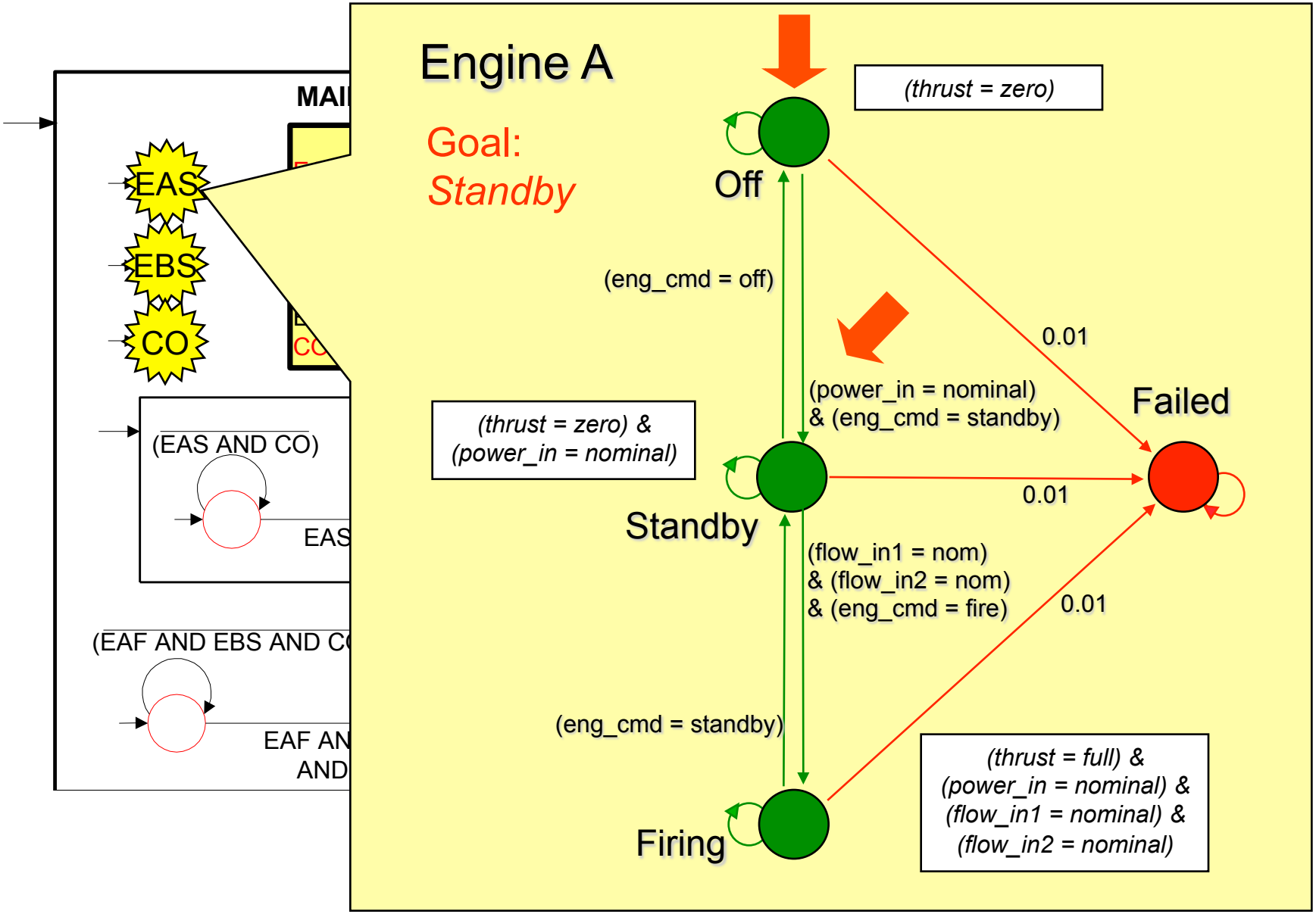




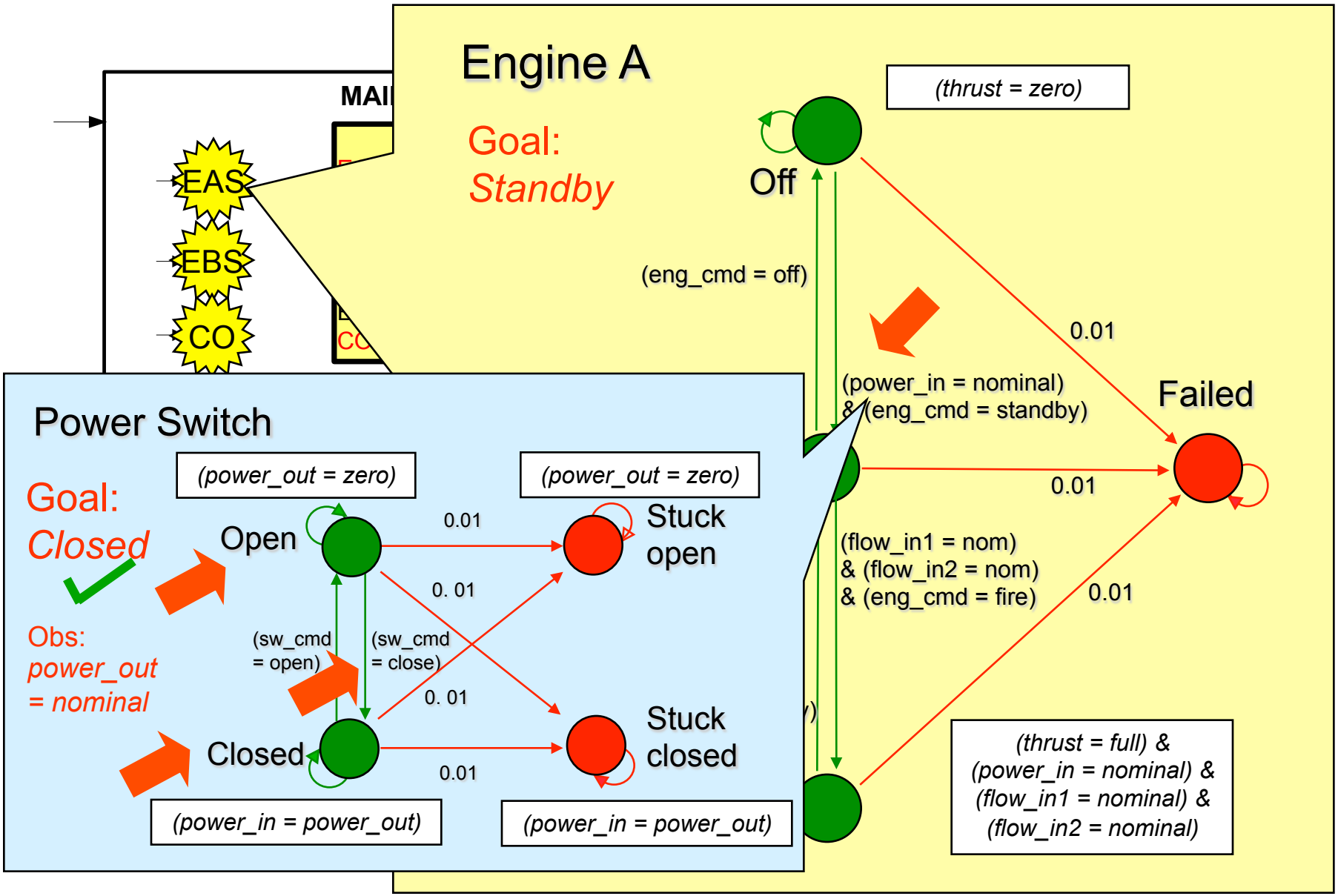
Deductive Controller estimates state and issues commands to achieve goals



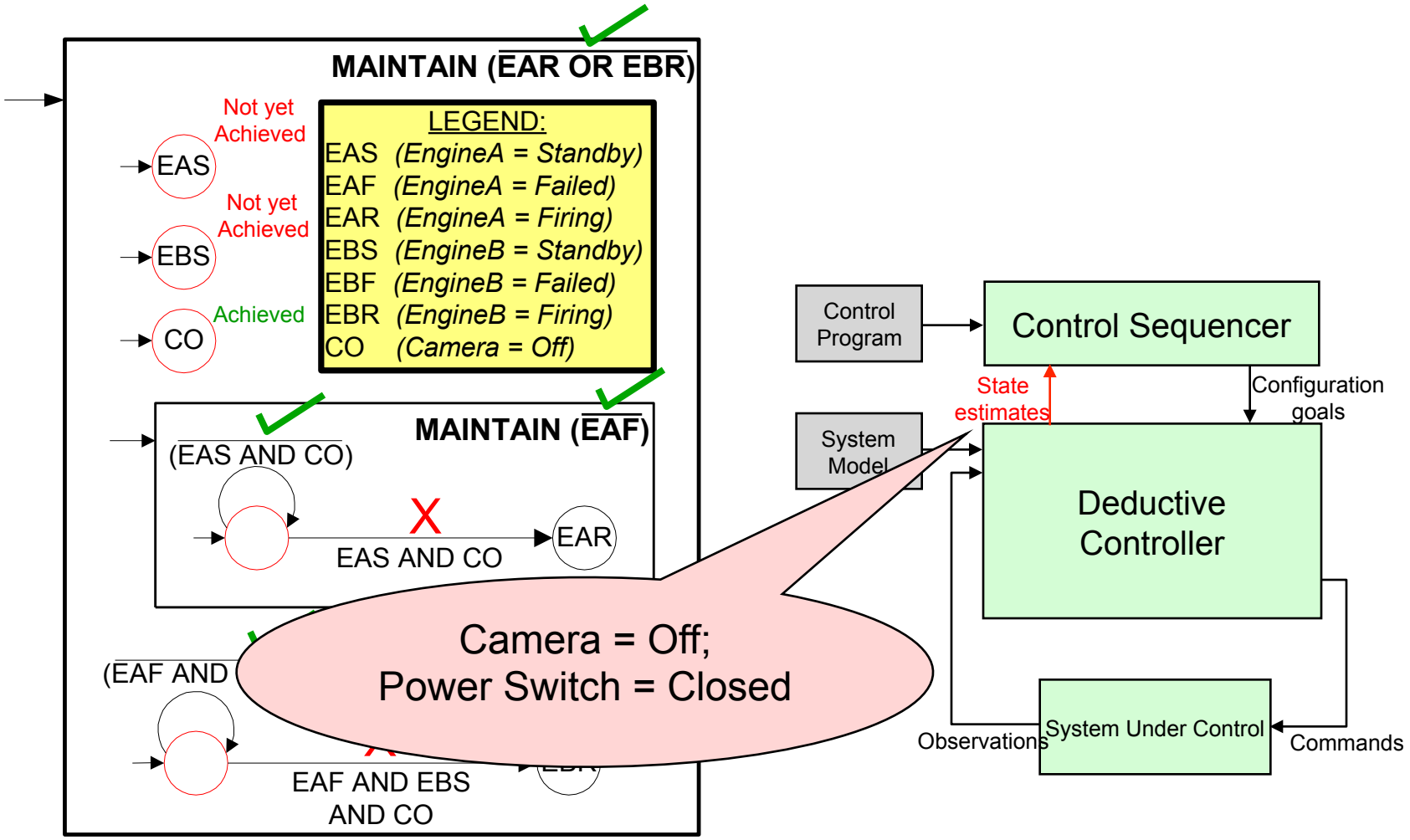
Deductive Controller estimates state and issues commands to achieve goals



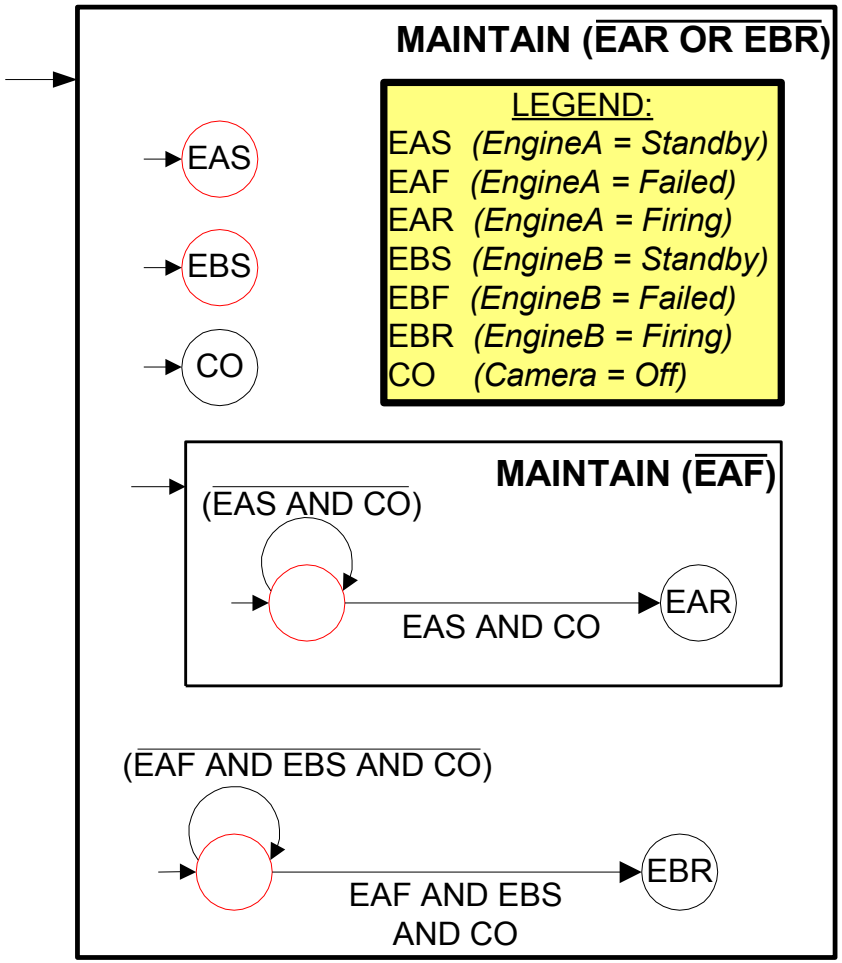
Deductive Controller estimates state and issues commands to achieve goals

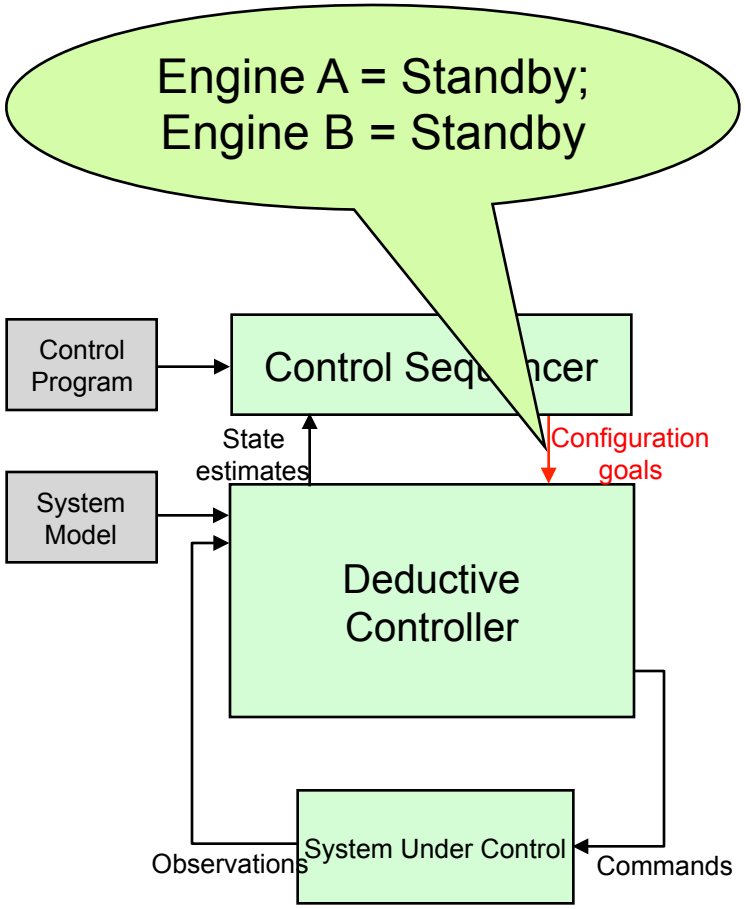


Executing HCA - Step 1



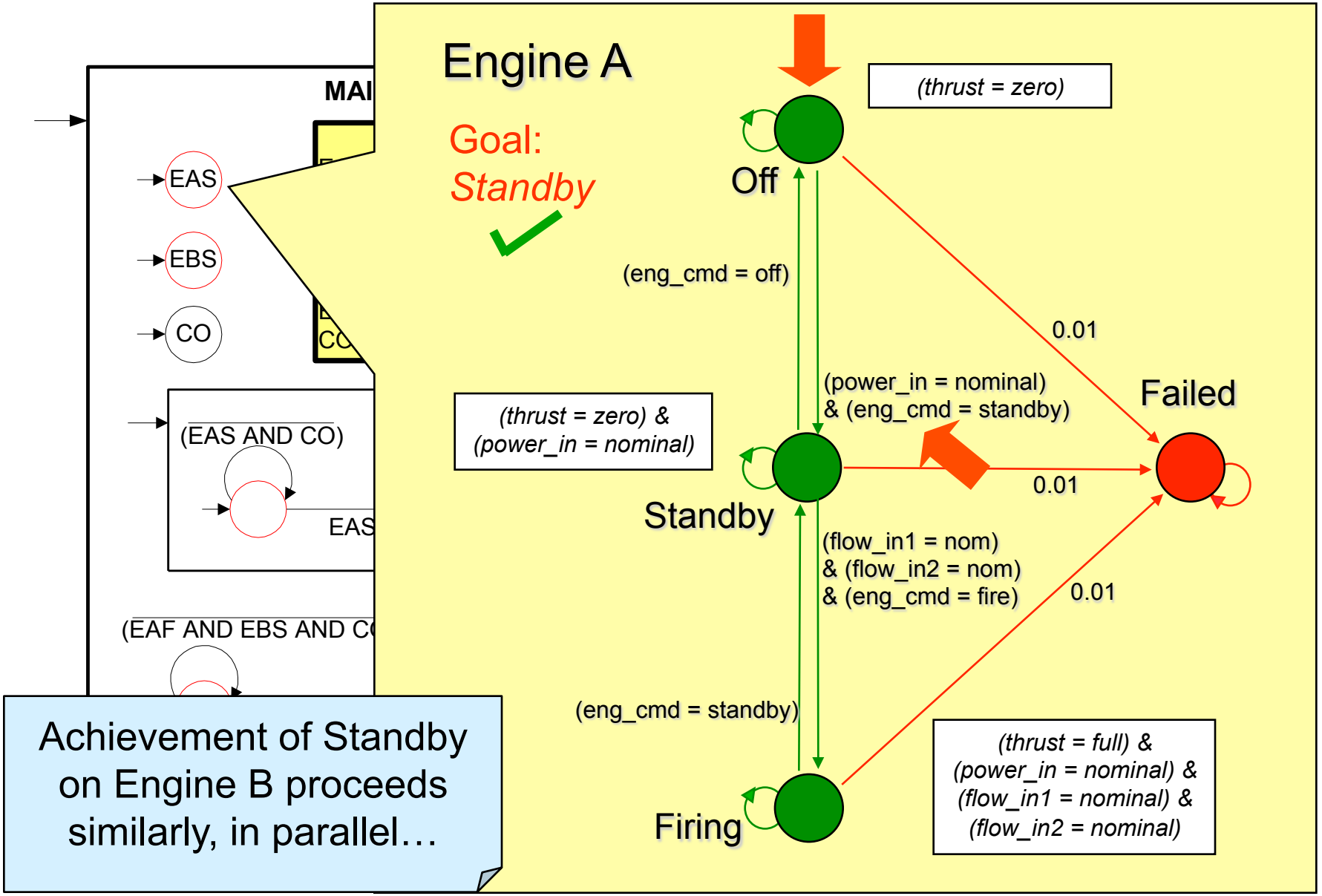
Executing HCA - Step 2





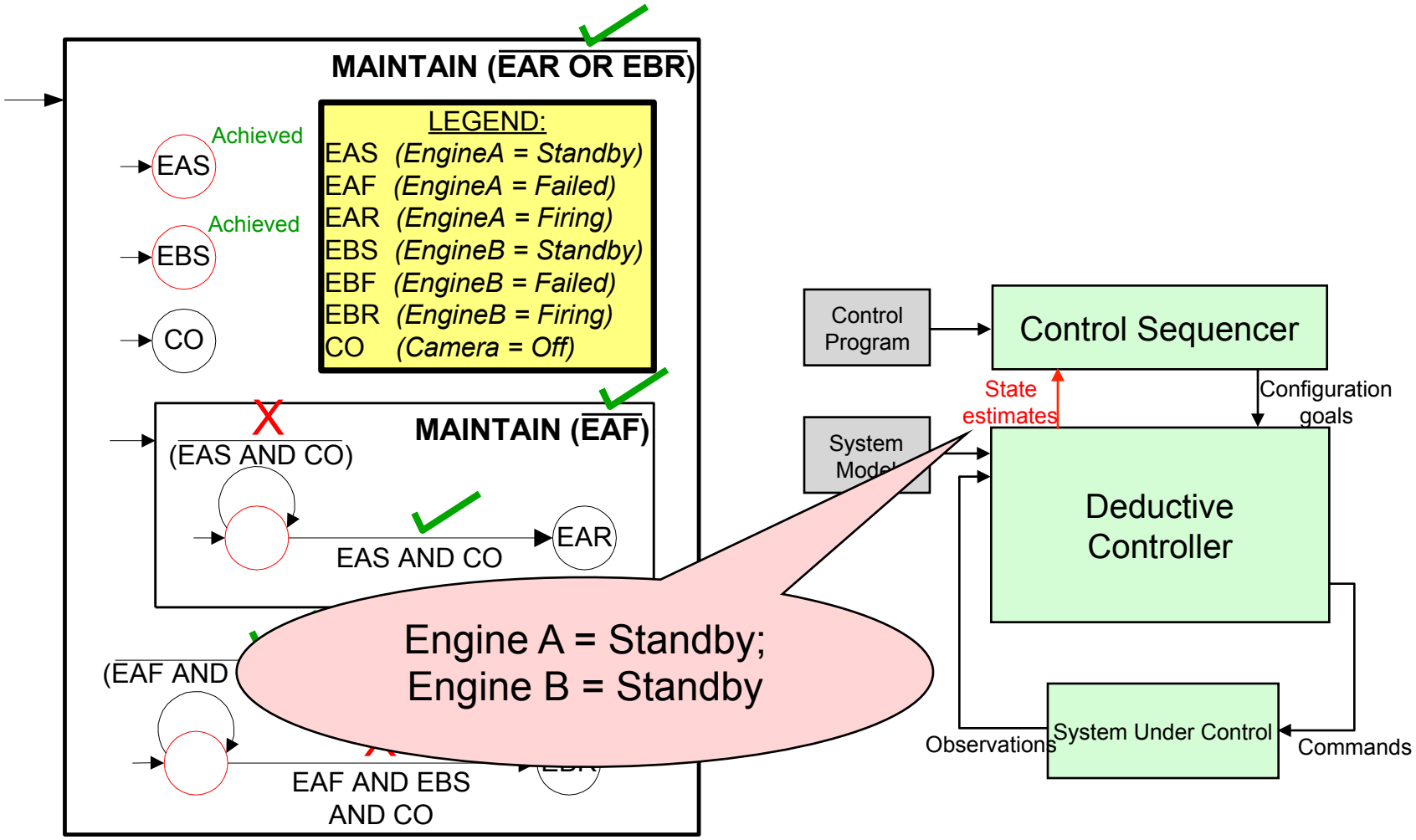
And so on…

Deductive Controller estimates state and issues commands to achieve goals

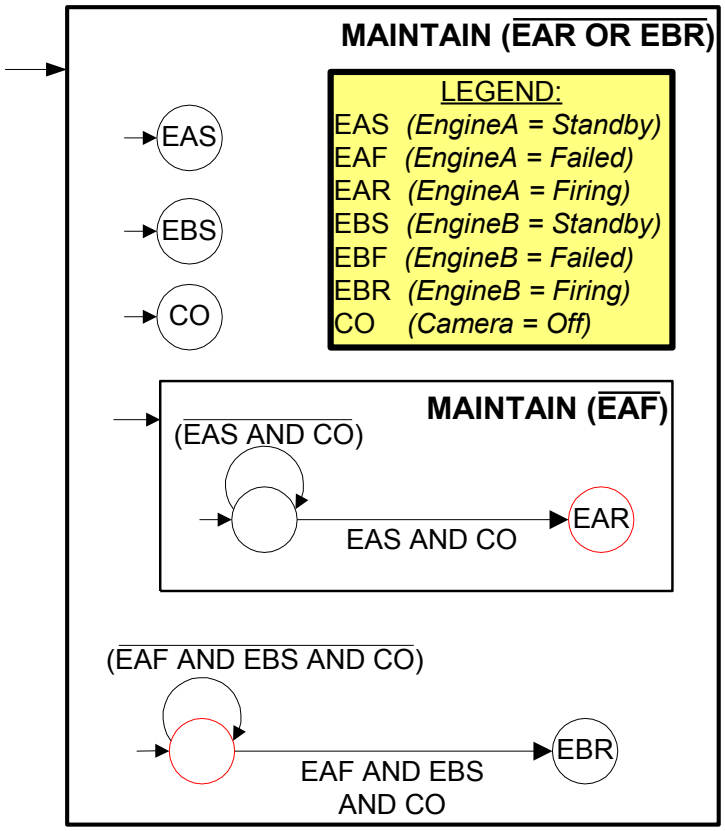


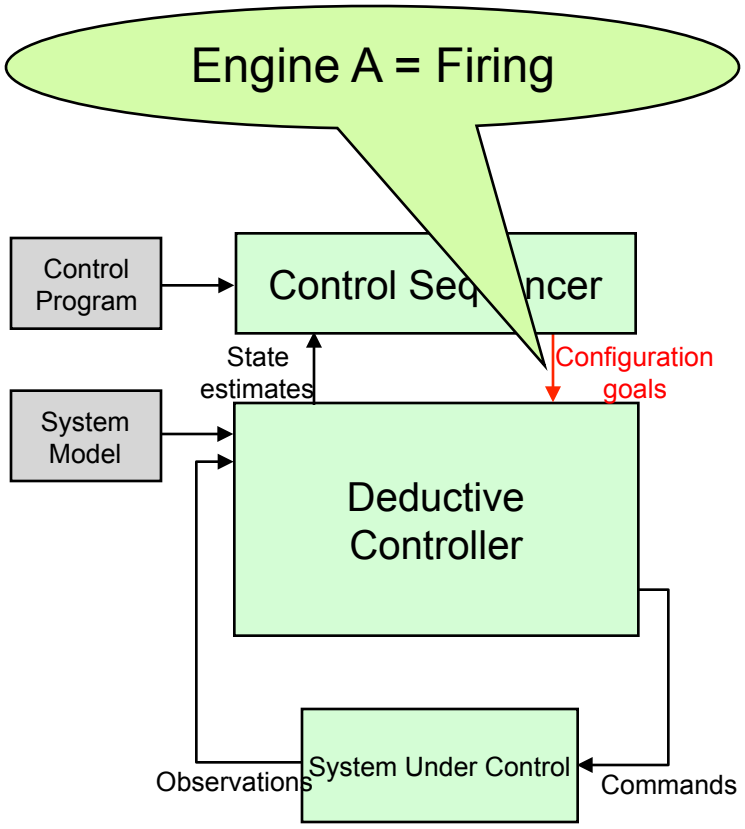
Achievement of Standby on Engine B proceeds similarly, in parallel…

Executing HCA - Step 2

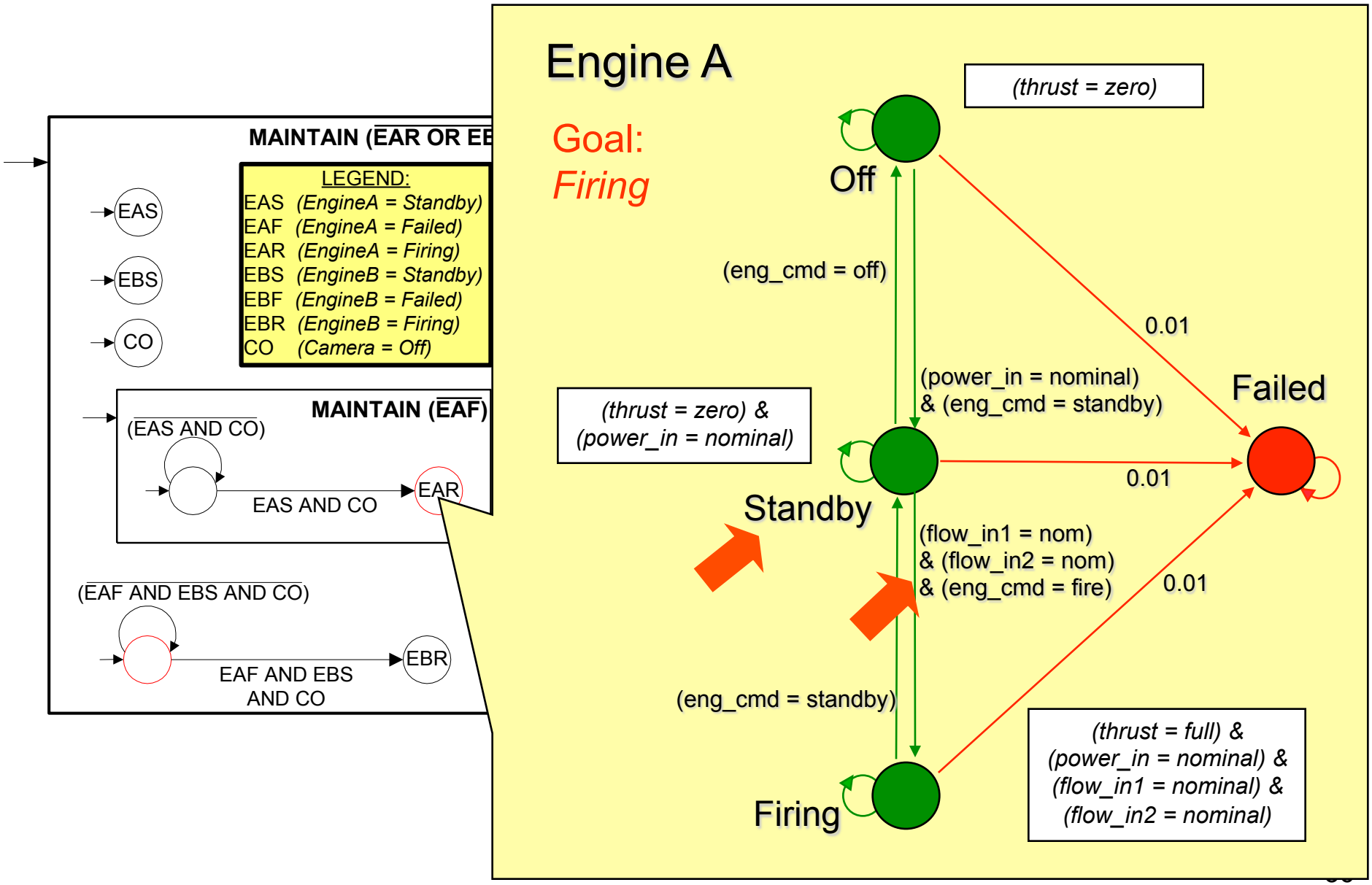


Executing HCA - Step 3

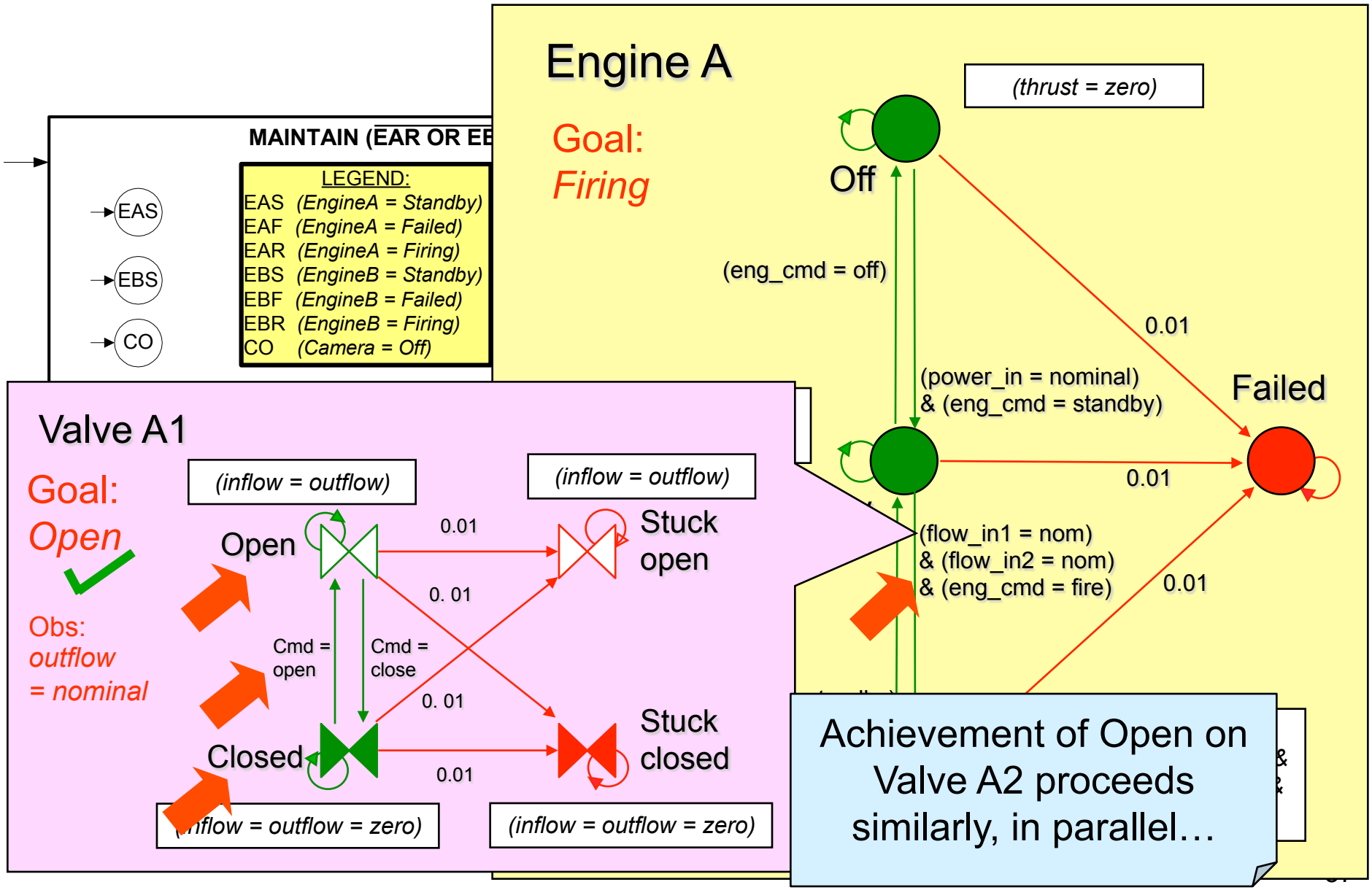




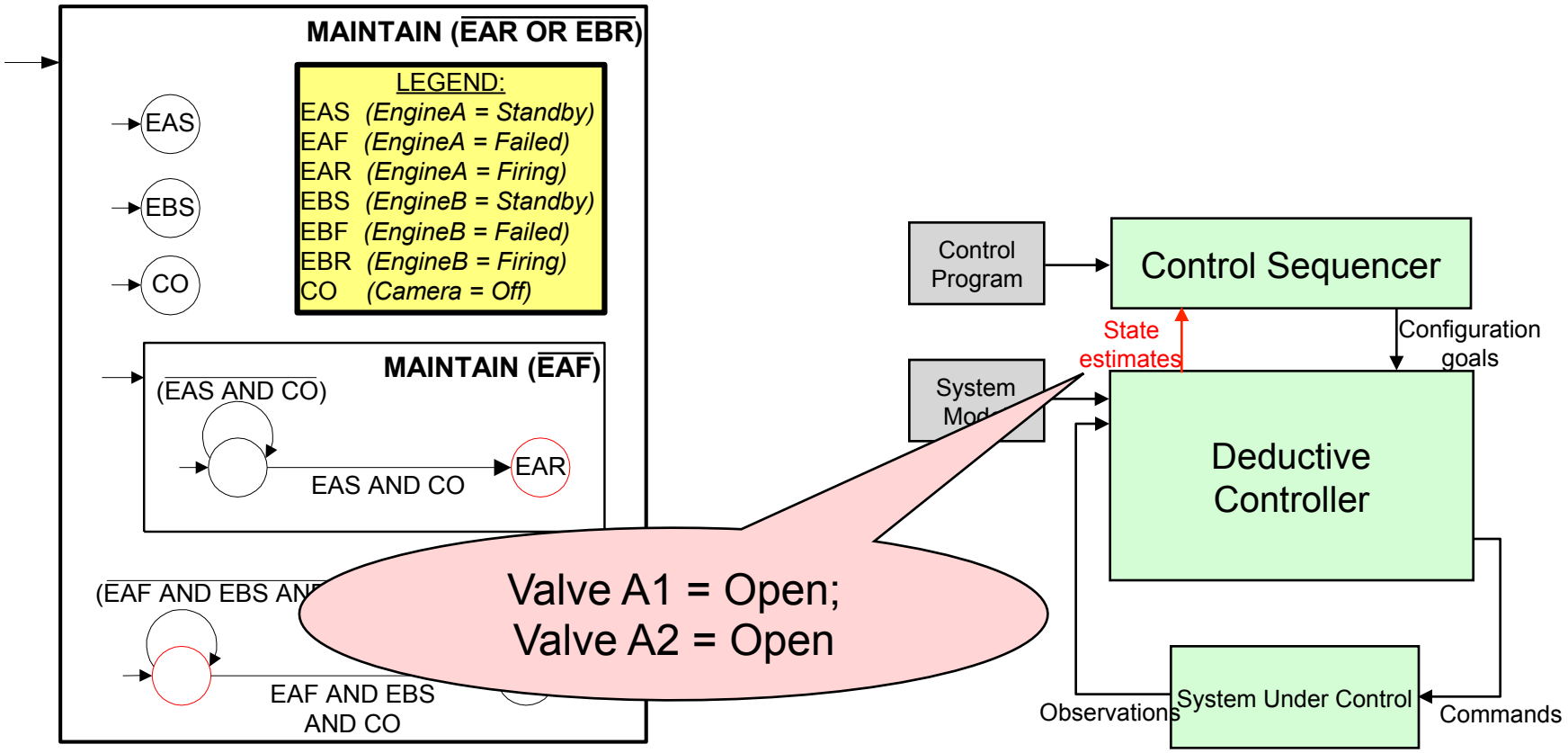
Deductive Controller estimates state and issues commands to achieve goals



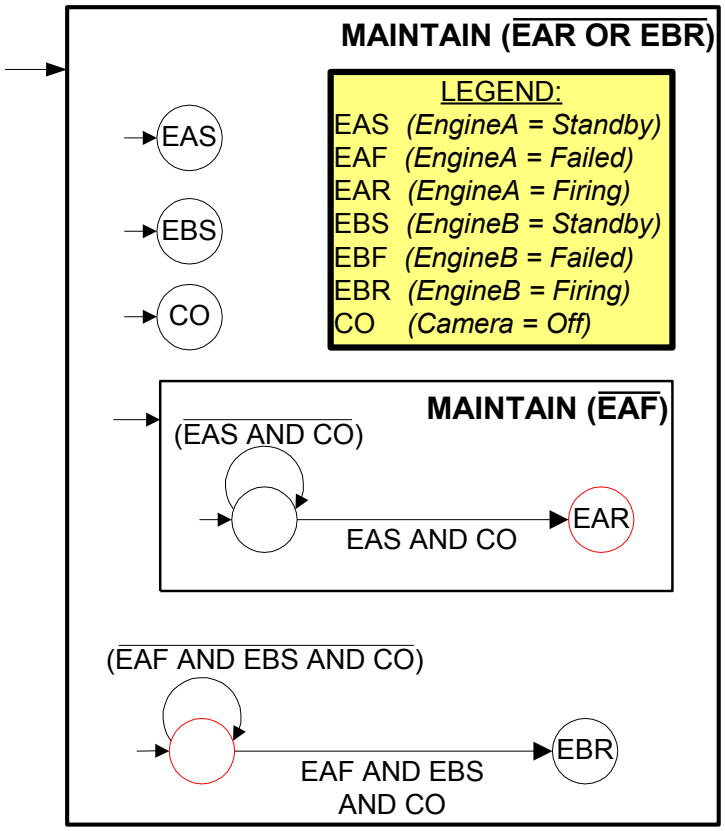
Deductive Controller estimates state and issues commands to achieve goals

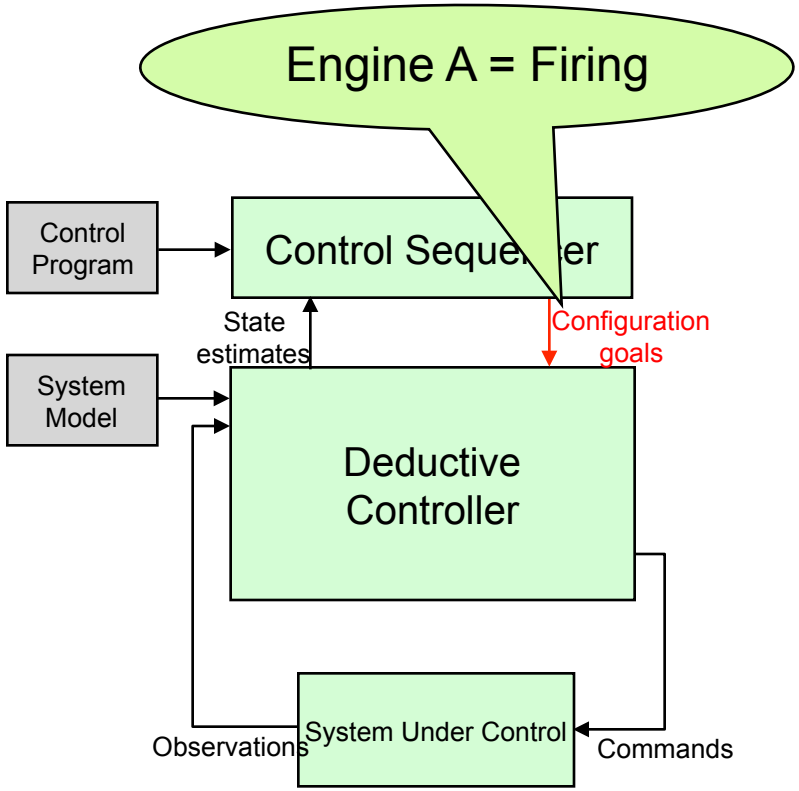


Executing HCA - Step 3

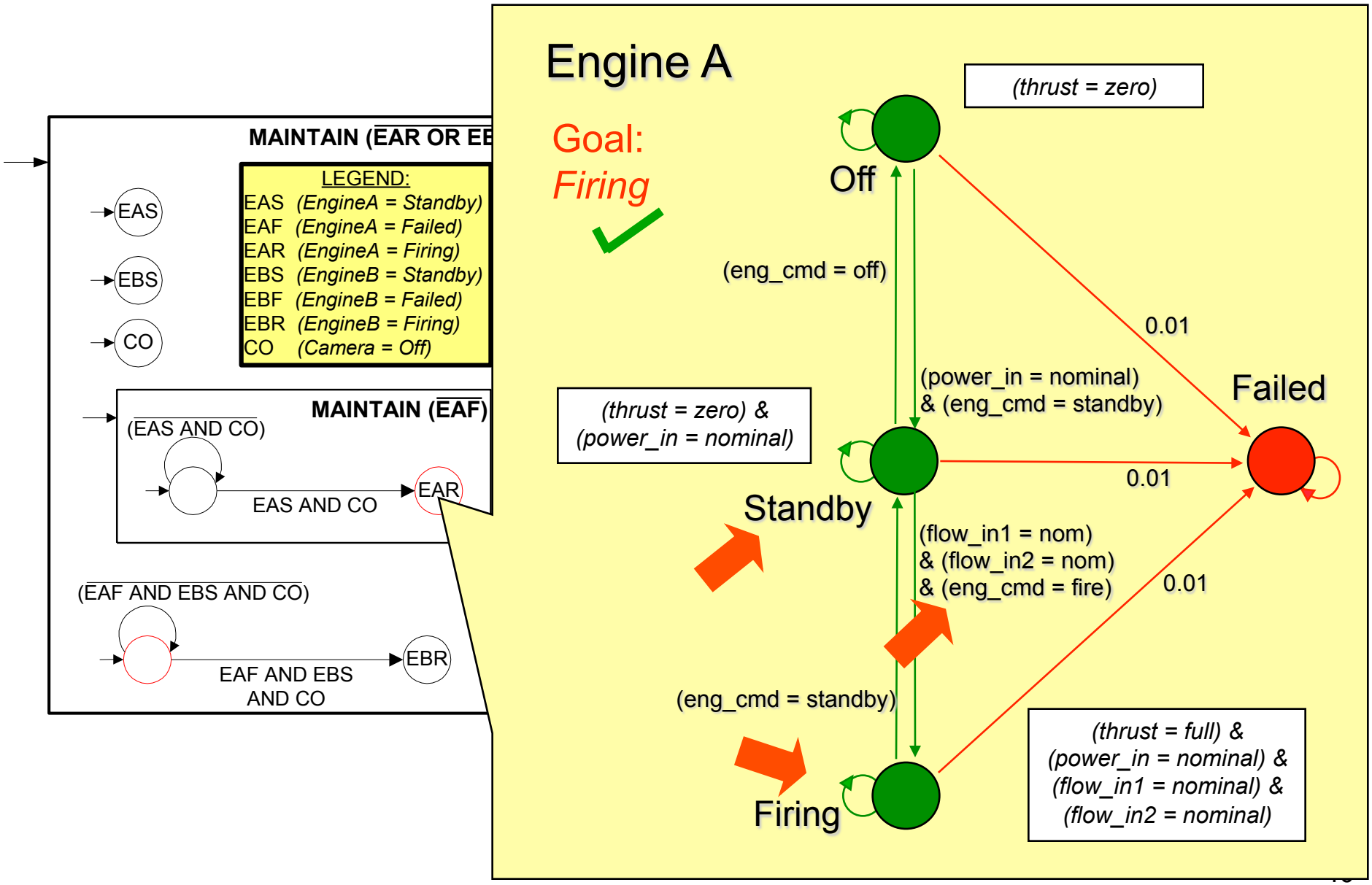


Executing HCA - Step 4

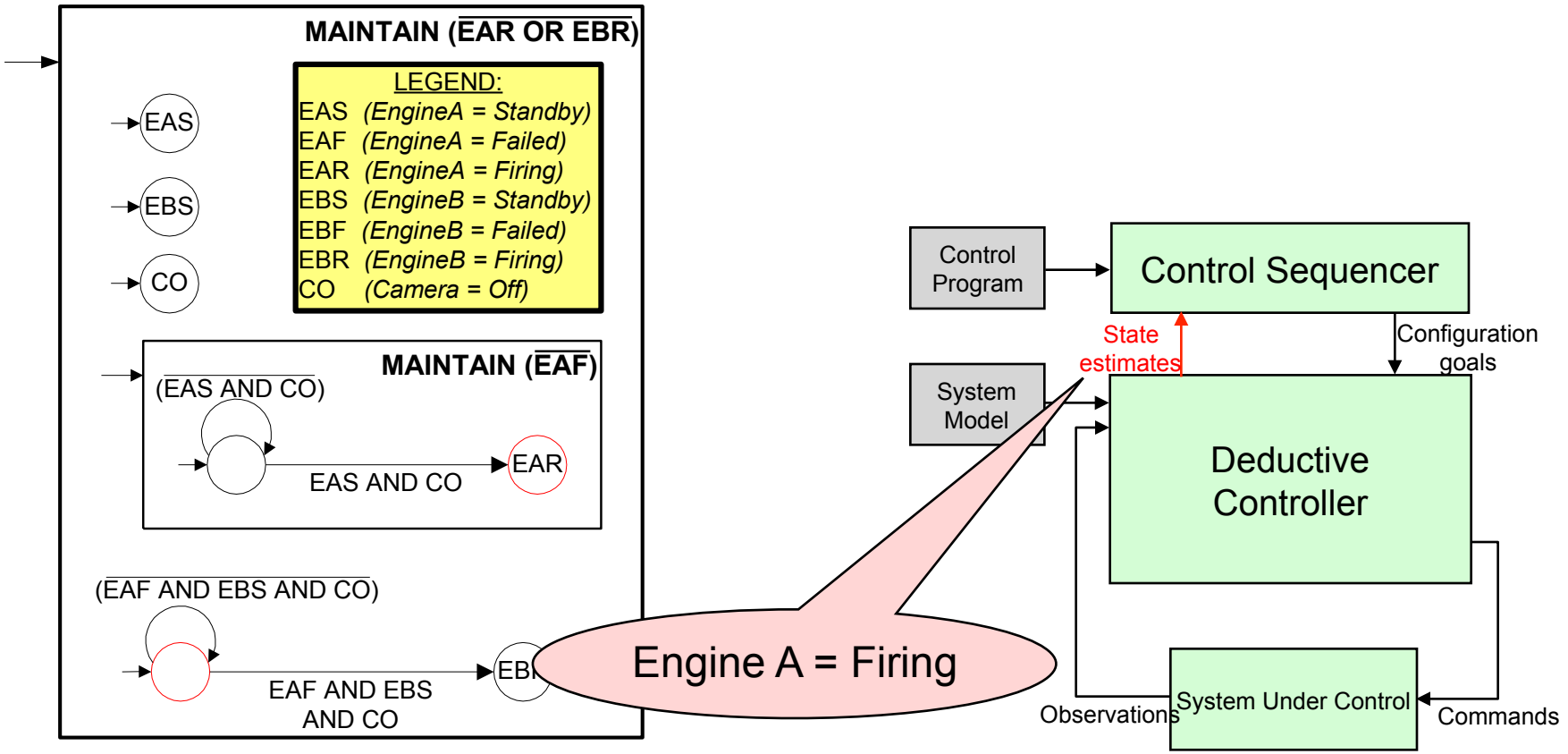




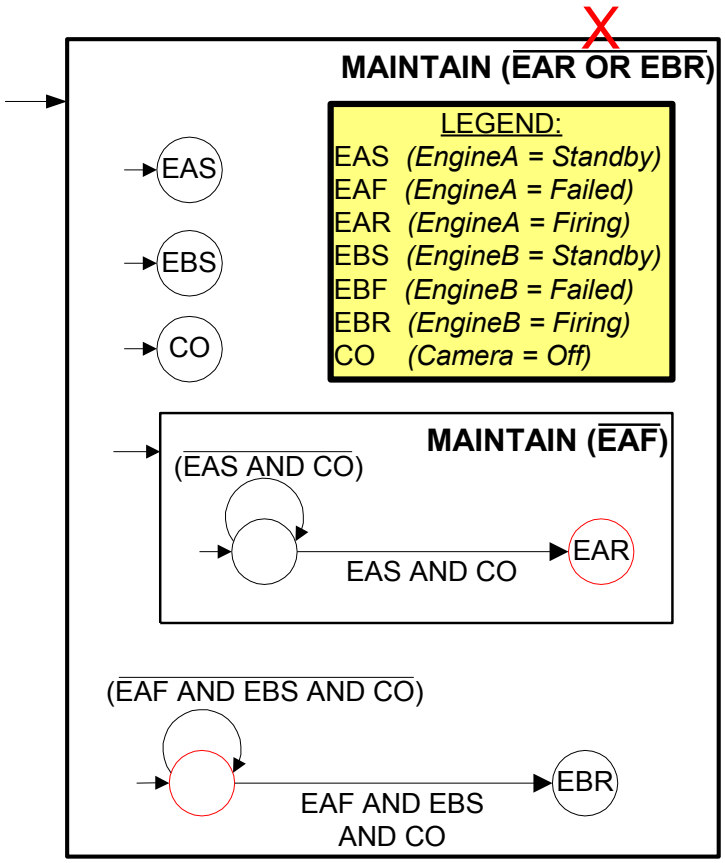
Deductive Controller estimates state and issues commands to achieve goals



Executing HCA - Step 4

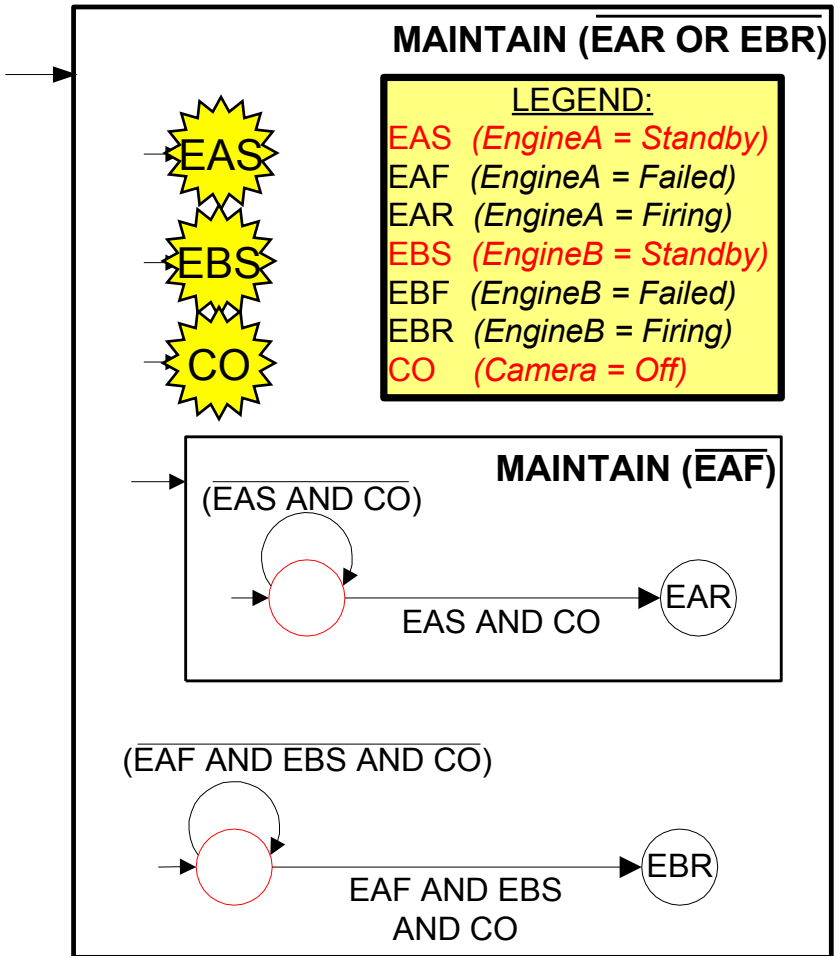


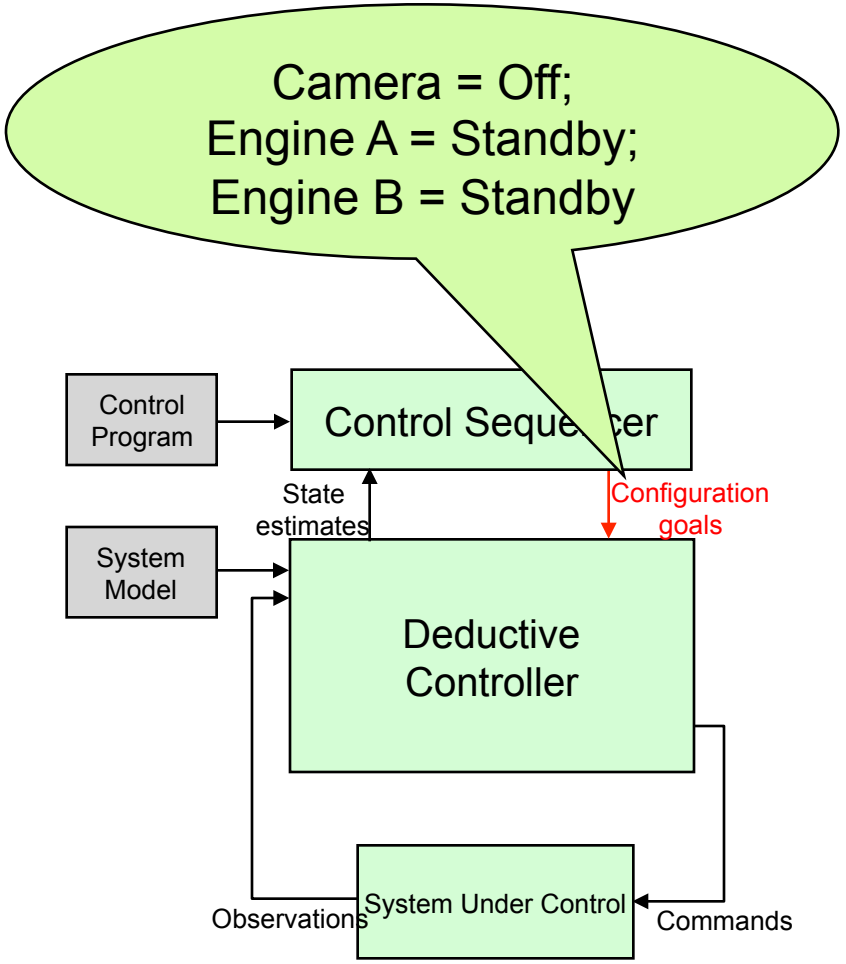
Executing HCA - Step 5



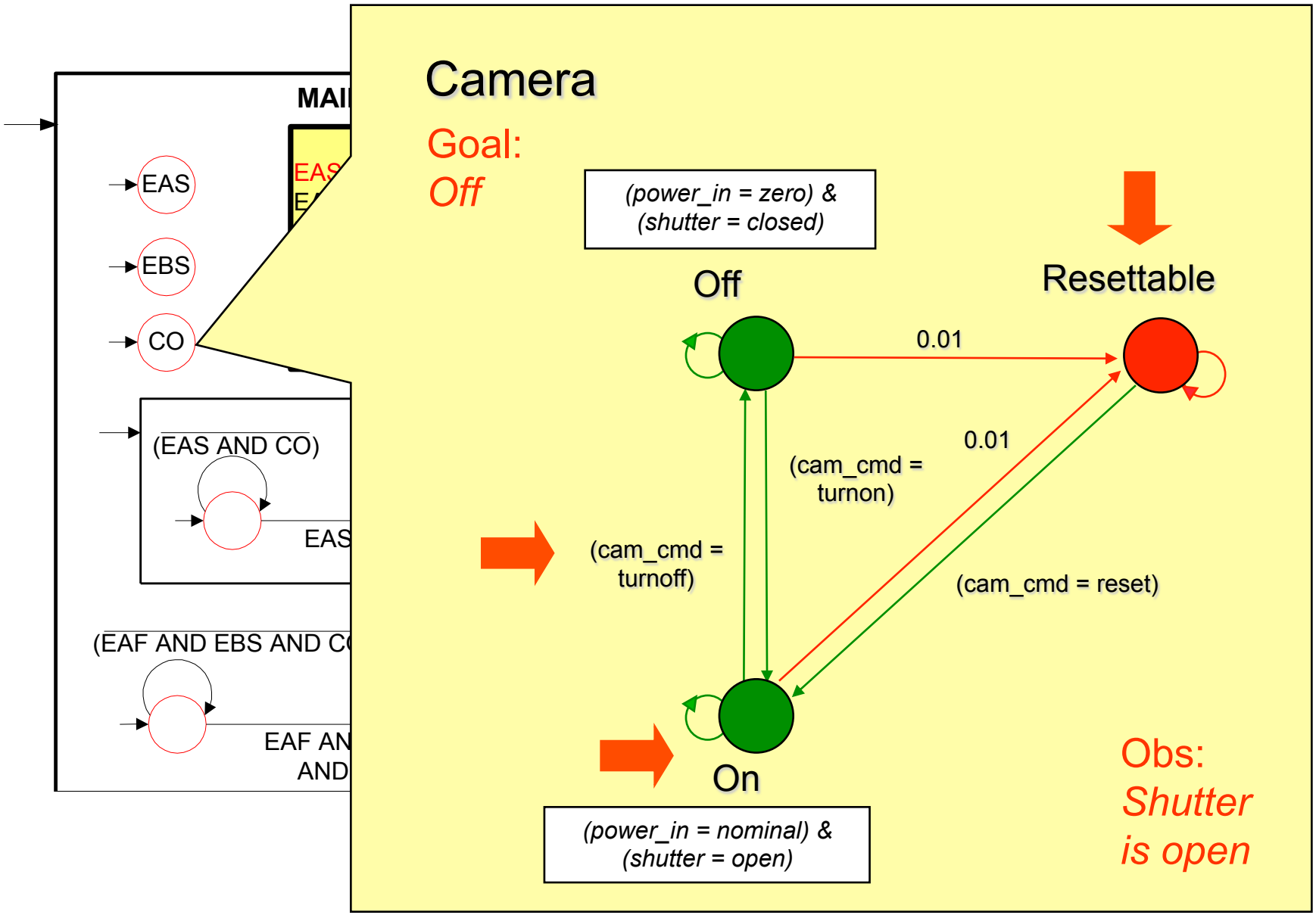
- *(EngineA = Firing)* achieved in this step
- maintenance condition violated, HCA block exited

What About Off-nominal Execution?

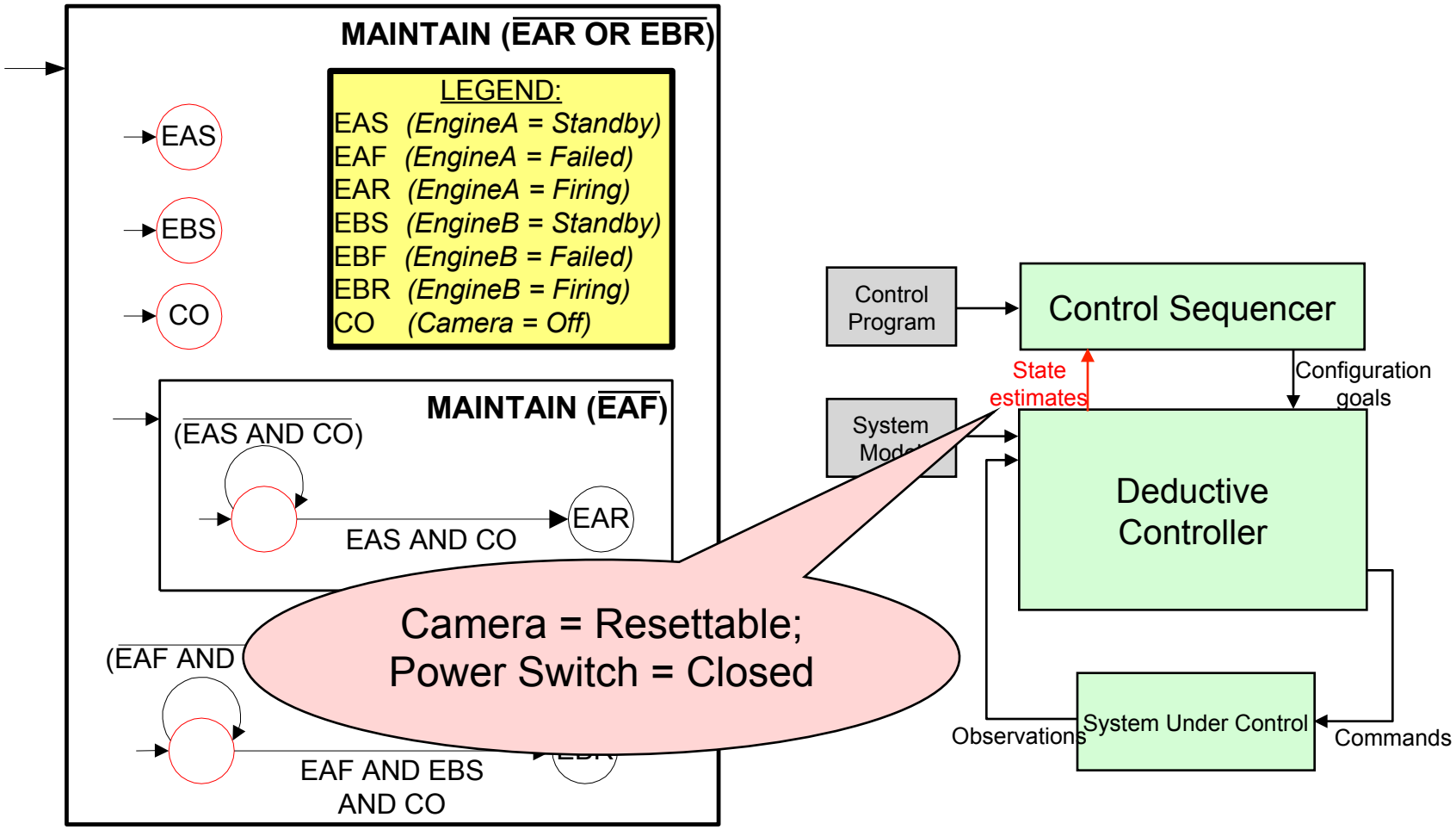




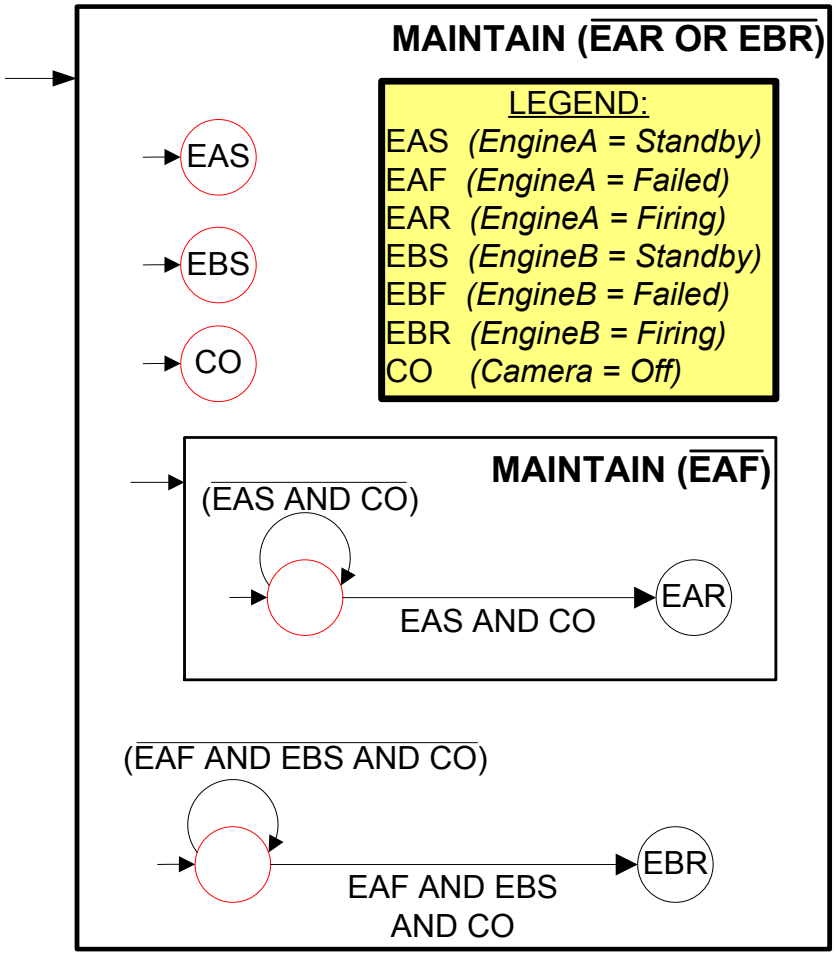
Model-based executive provides in-the-loop robustness

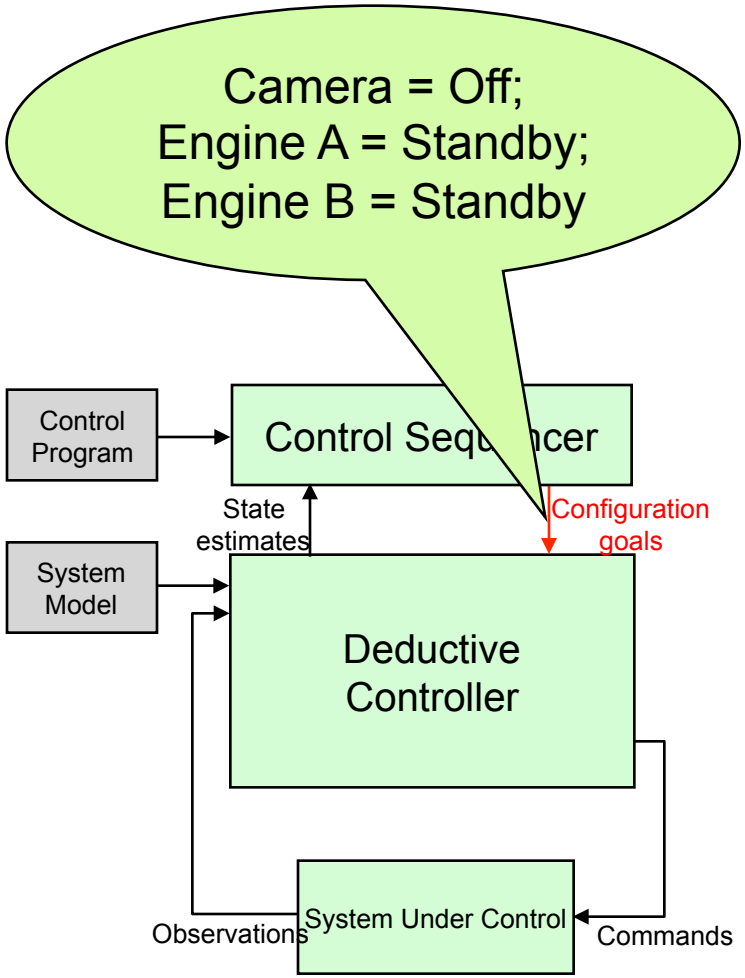


Executing HCA - Step 1

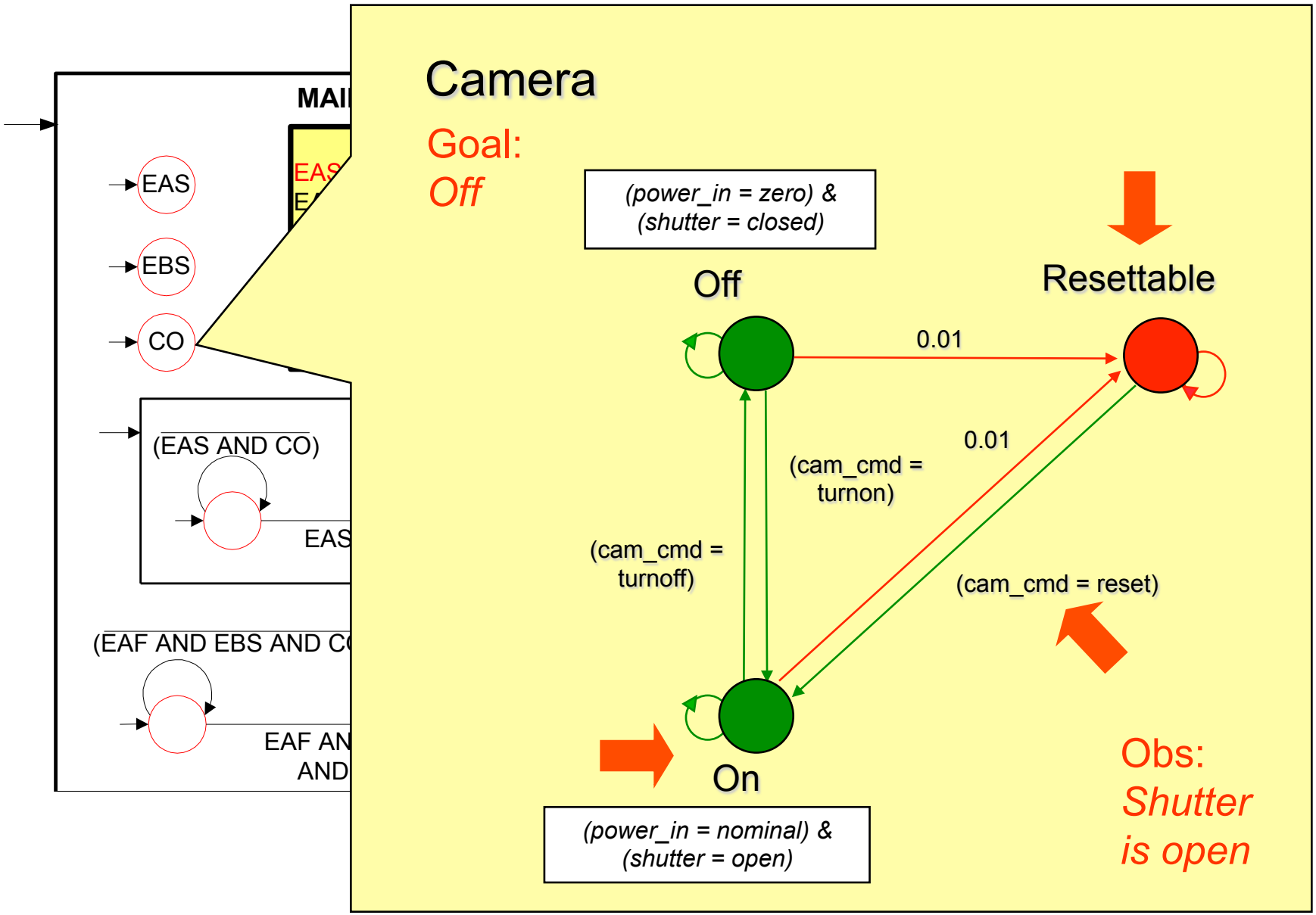


Executing HCA - Step 2

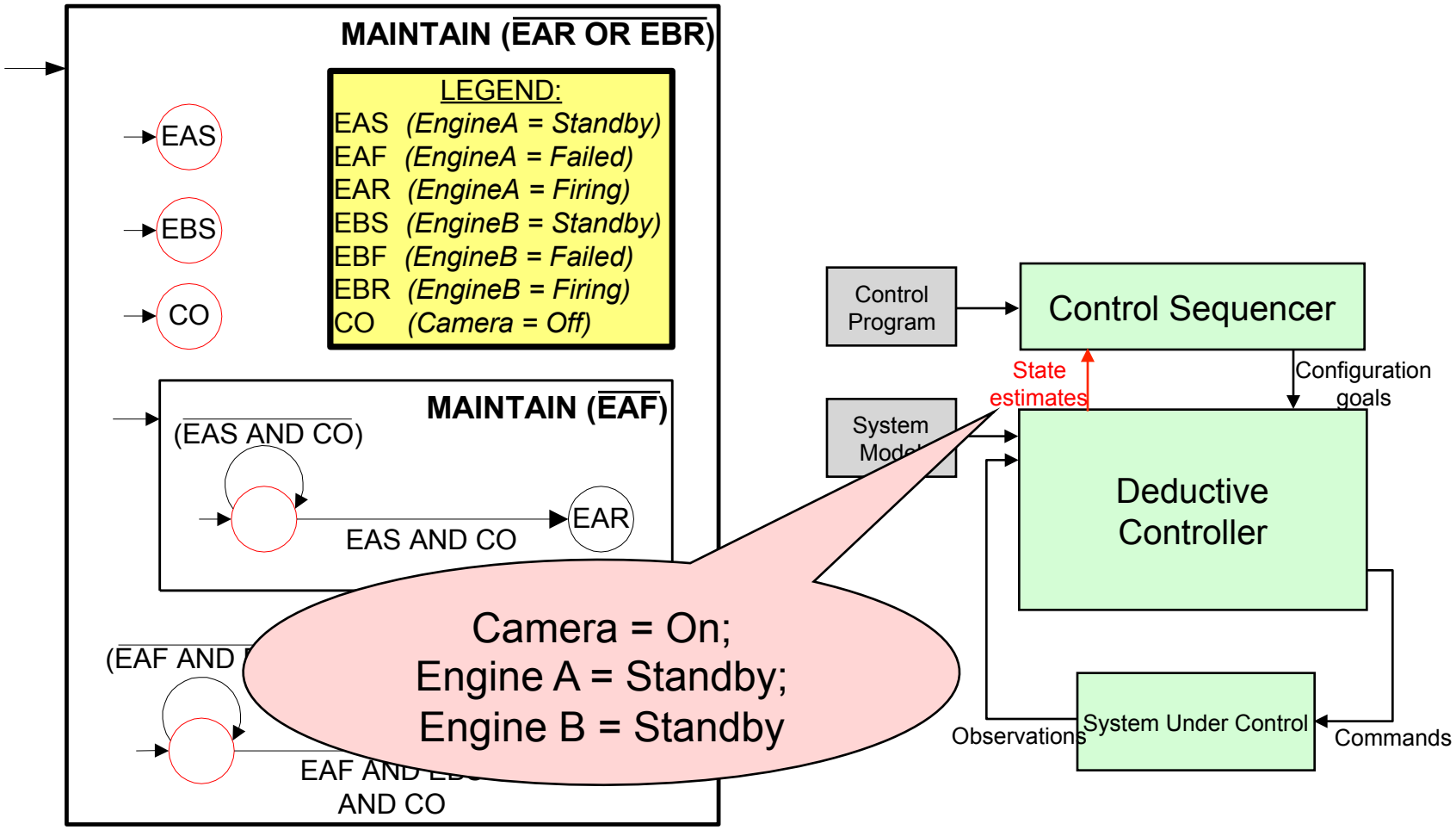




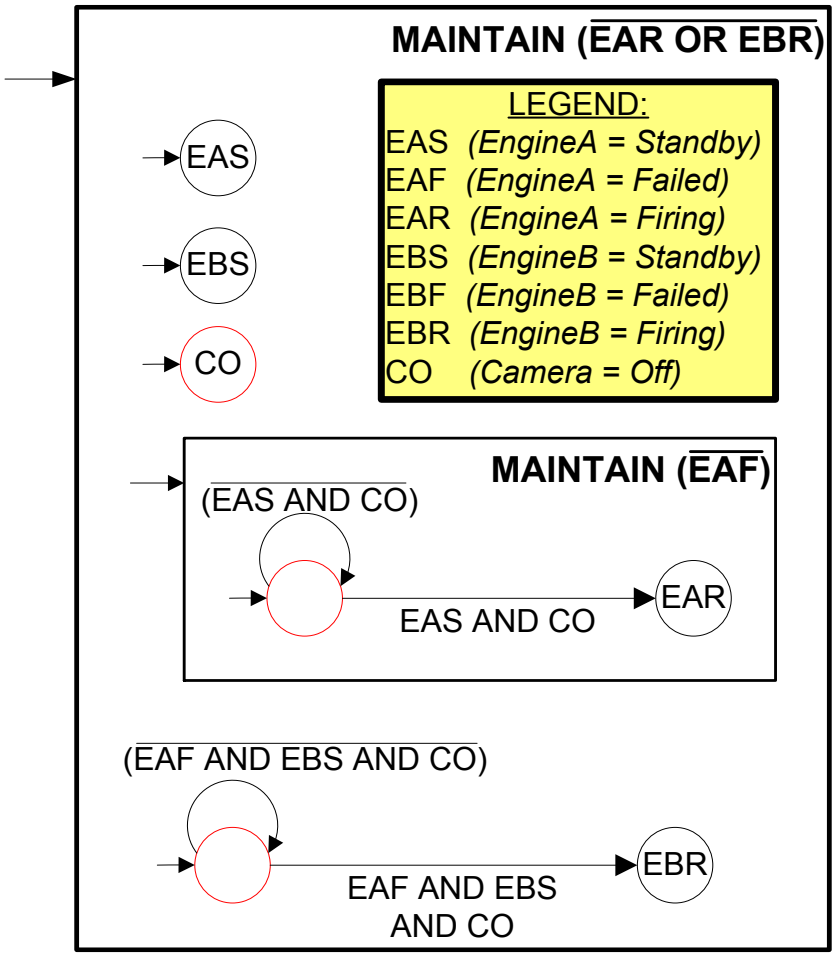
Model-based executive provides in-the-loop robustness

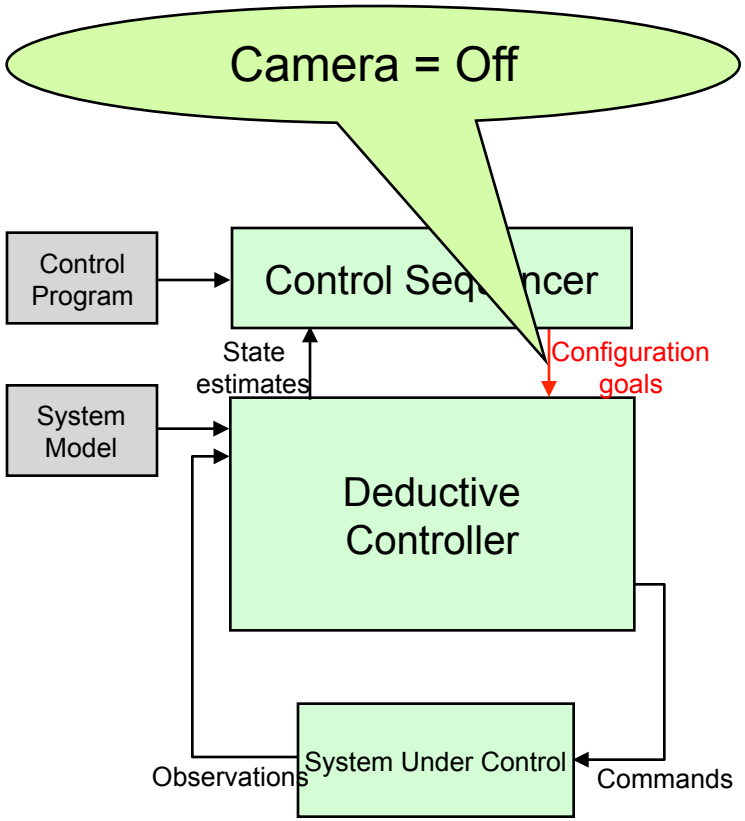


Executing HCA - Step 2

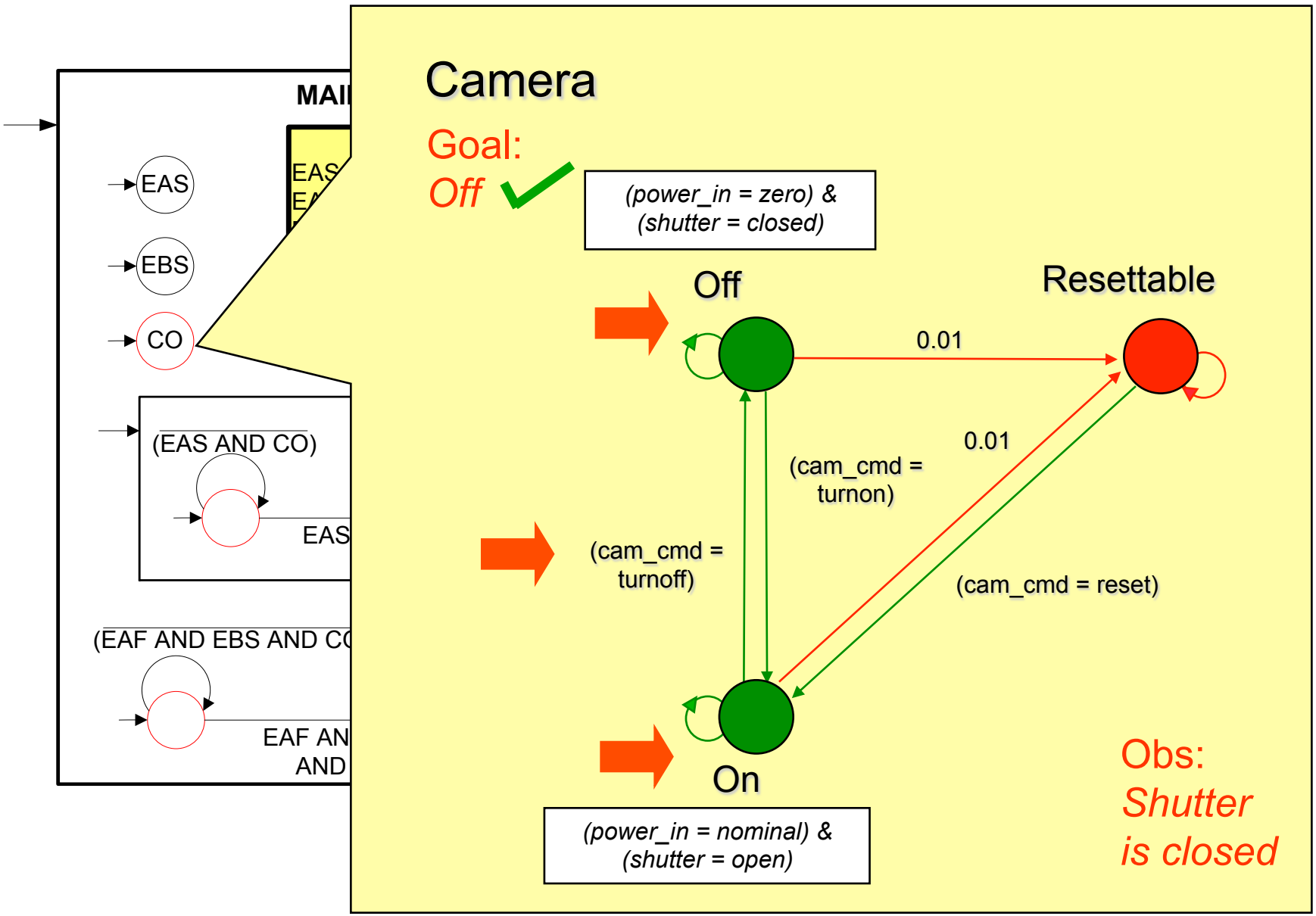


Executing HCA - Step 3

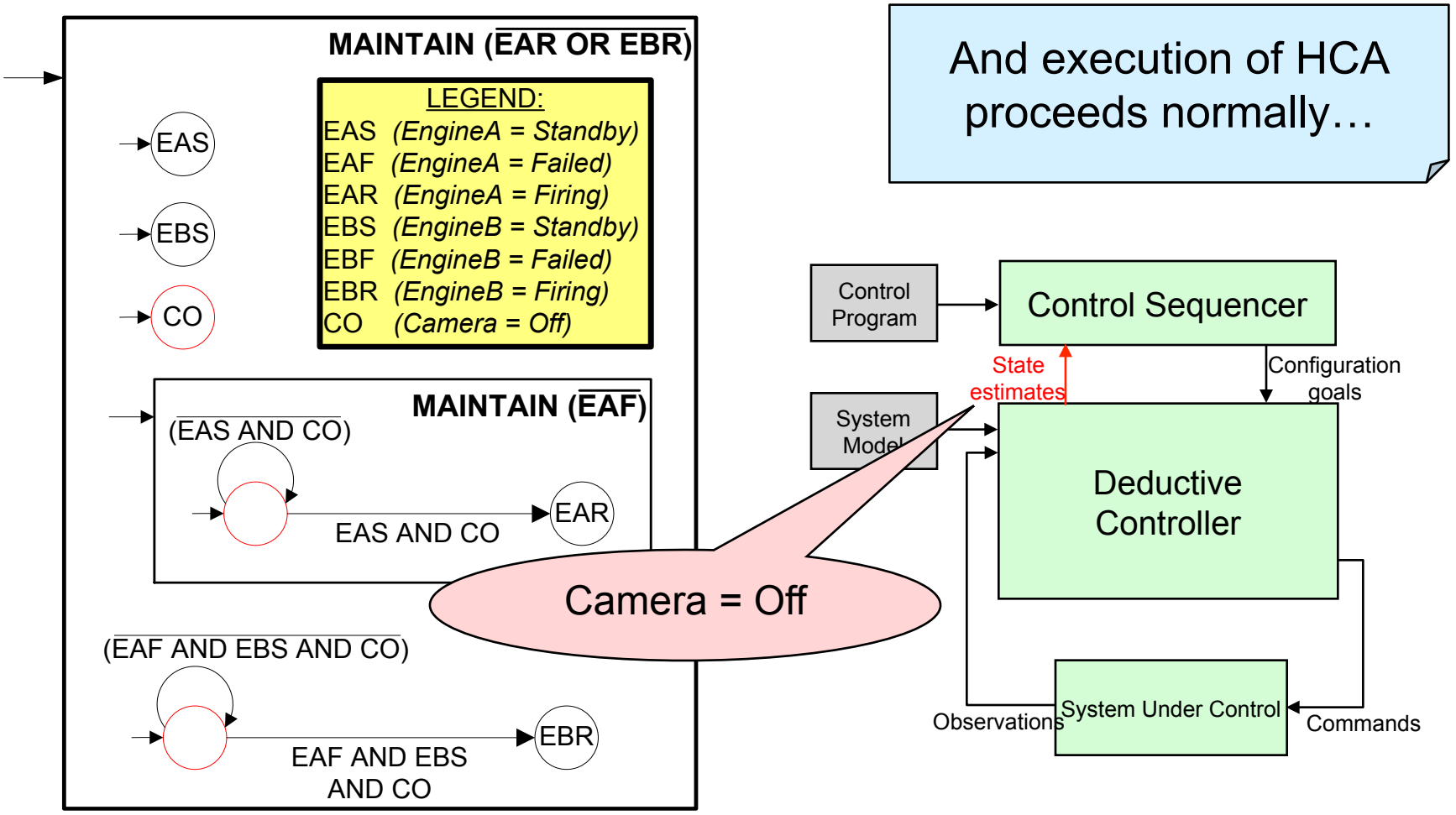




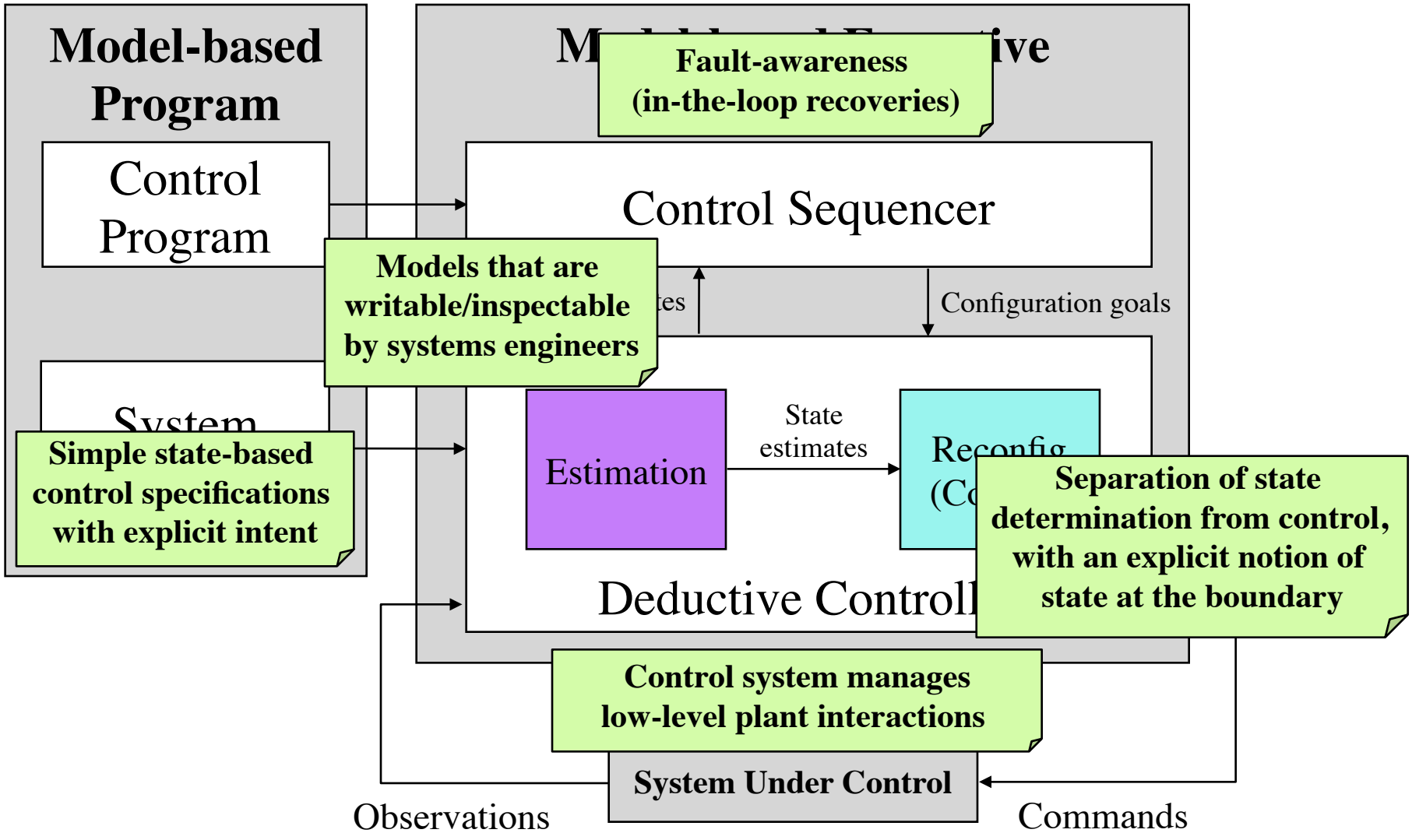
Model-based executive provides in-the-loop robustness



Executing HCA - Step 3



Desirable Architectural Features, Revisited



Key Take-Away Points

- Single control architecture for *both nominal and off-nominal* execution
  - Off-nominal situations handled at different layers in the architecture, as appropriate.
  - Design trade: in which layer will a given failure be handled?
- State is central and intent is *explicit*
- Models are used *directly* in each layer of the architecture
- Details of the implementation are less important than the architectural features (*principles*)
  - A variant of this architecture could employ more traditional estimation and control techniques/algorithms, and still get benefit.

Excerpt from Planetary Science Decadal Survey

New Frontiers 4 Selection

- Select NF-4 from among:
  - *Comet Surface Sample Return*
  - *Lunar South Pole-Aitken Basin Sample Return*
  - *Saturn Probe*
  - *Trojan Tour and Rendezvous*
  - *Venus In Situ Explorer*
- No relative priorities among these are assigned.
- If the selected NF-3 mission addresses the goals of one of these, remove that one from the list.

Excerpt from Steve Squyres presentation
'Vision and Voyages For Planetary Science in the Decade 2013-2022'

Excerpt from Planetary Science Decadal Survey

Flagship Missions

(in priority order)

1. Begin NASA/ESA Mars Sample Return campaign: *Descoped Mars Astrobiology Explorer-Cacher (MAX-C)*
2. Detailed investigation of a probable ocean in the outer solar system: *Descoped Jupiter Europa Orbiter (JEO)*
3. First in-depth exploration of an Ice Giant planet: *Uranus Orbiter and Probe*
4. Either *Enceladus Orbiter* or *Venus Climate Mission* (no relative priorities assigned)

Excerpt from Steve Squyres presentation
'Vision and Voyages For Planetary Science in the Decade 2013-2022'

28

NATIONAL RESEARCH COUNCIL
OF THE NATIONAL ACADEMIES

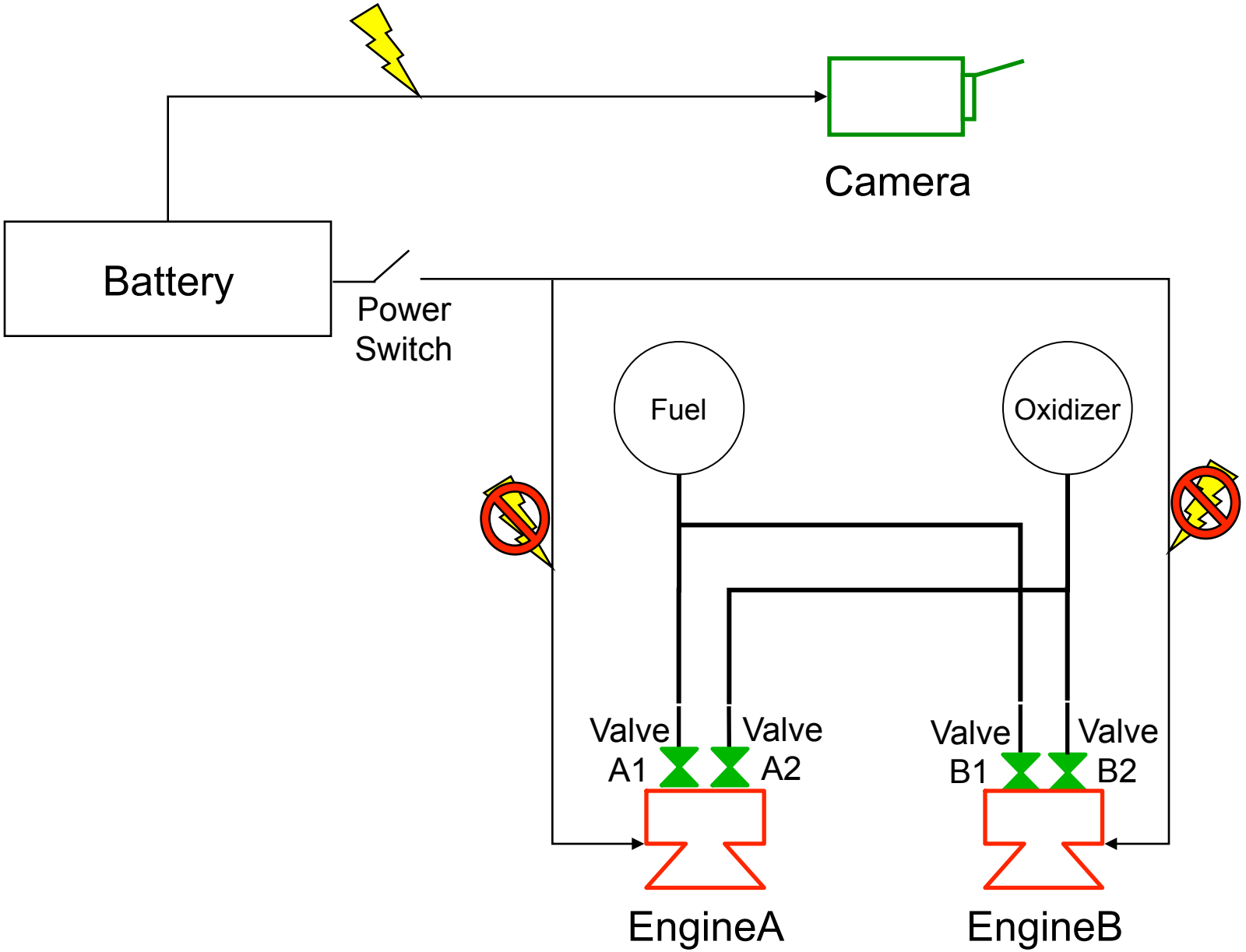
For More Information

- Williams, B.C., Ingham, M.D., Chung, S.H., and Elliott, P.H., "Model-based Programming of Intelligent Embedded Systems and Robotic Space Explorers", *Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software*, Vol. 91, No. 1, Jan. 2003, pp. 212-237.

- "A Short Course in Model-based Autonomy", presented at the 2005 IEEE Conference on Systems, Man, and Cybernetics, Oct. 9th, 2005, Waikoloa, HI (co-instructors: B. Williams, P. Robertson).

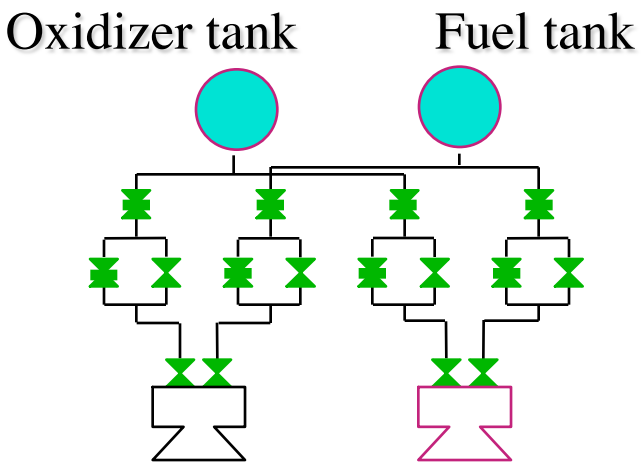
BACKUP

The System Under Control



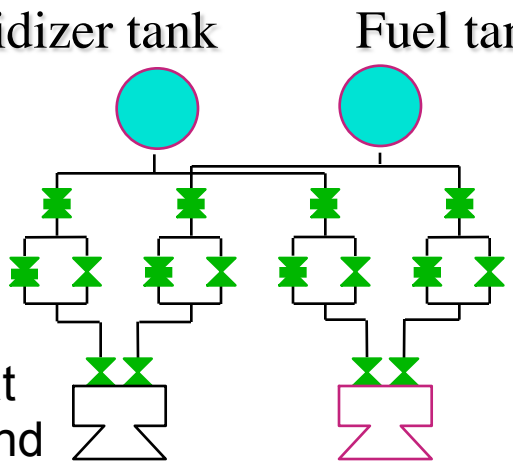
Example: Model-based Executive

- States like *(EngineA = Firing)* are not necessarily DIRECTLY observable or controllable
- When the Control Sequencer issues the configuration goal *(EngineA = Firing)*, the Deductive Controller…



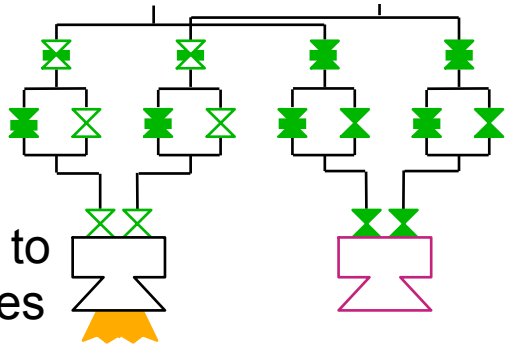
Example: Model-based Executive

Mode Estimation



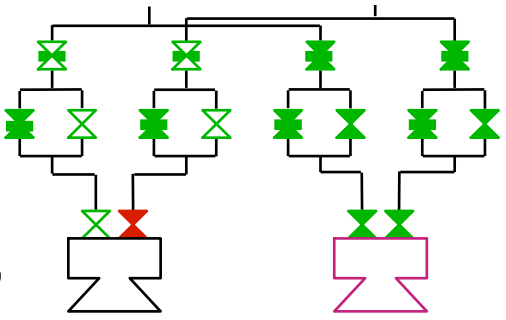
Deduces that thrust is off, and the engine is healthy

Mode Reconfiguration



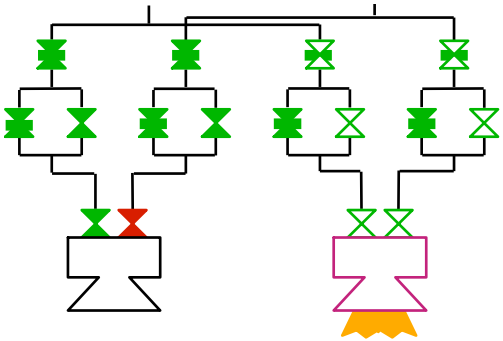
Plans actions to open six valves and executes them, one at a time

Deduces that a valve failed - stuck closed



Mode Estimation

Determines valves on the backup engine that will achieve thrust, plans needed actions and executes them.



Mode Reconfiguration