

PLASMA

Plan State Management Architecture

**A Planning and Scheduling
Component Library**

for

NASA

Missions and Research

Team & Contributors

Andrew Bachmann, Tania Bedrax-Weiss, Patrick Daley, Will Edgington, Jeremy Frank, Michael Iatauro, Ari Jonsson, Conor McGann (PI), Paul Morris, Sailesh Ramakrishnan, Will Taylor



PLASMA Motivation



NASA needs

- To solve large class of problems relevant to space exploration
- Requires wide variety of planning algorithms
- Requires advanced inference for many classes of constraints
- Requires integration into a wide variety of applications/architectures

Technology Components

- A powerful modeling language – to describe the problem domain
- A robust and powerful Plan Database – enforce plan consistency and infer consequences of plan modifications
- A Planner Application Framework – support integration

Benefits

- Increase capabilities of mission and research applications
- Reduce development cost and risk
- Encourage technology transfer through common infrastructure



A Variety of Uses Applications



Missions

- DS1: RAX – Remote Agent Experiment (original version of technology) '99
- Life in the Artacama (LITA) Desert Rover '04
- MER - Mars Exploration Rover Science Planning Tool '03-04

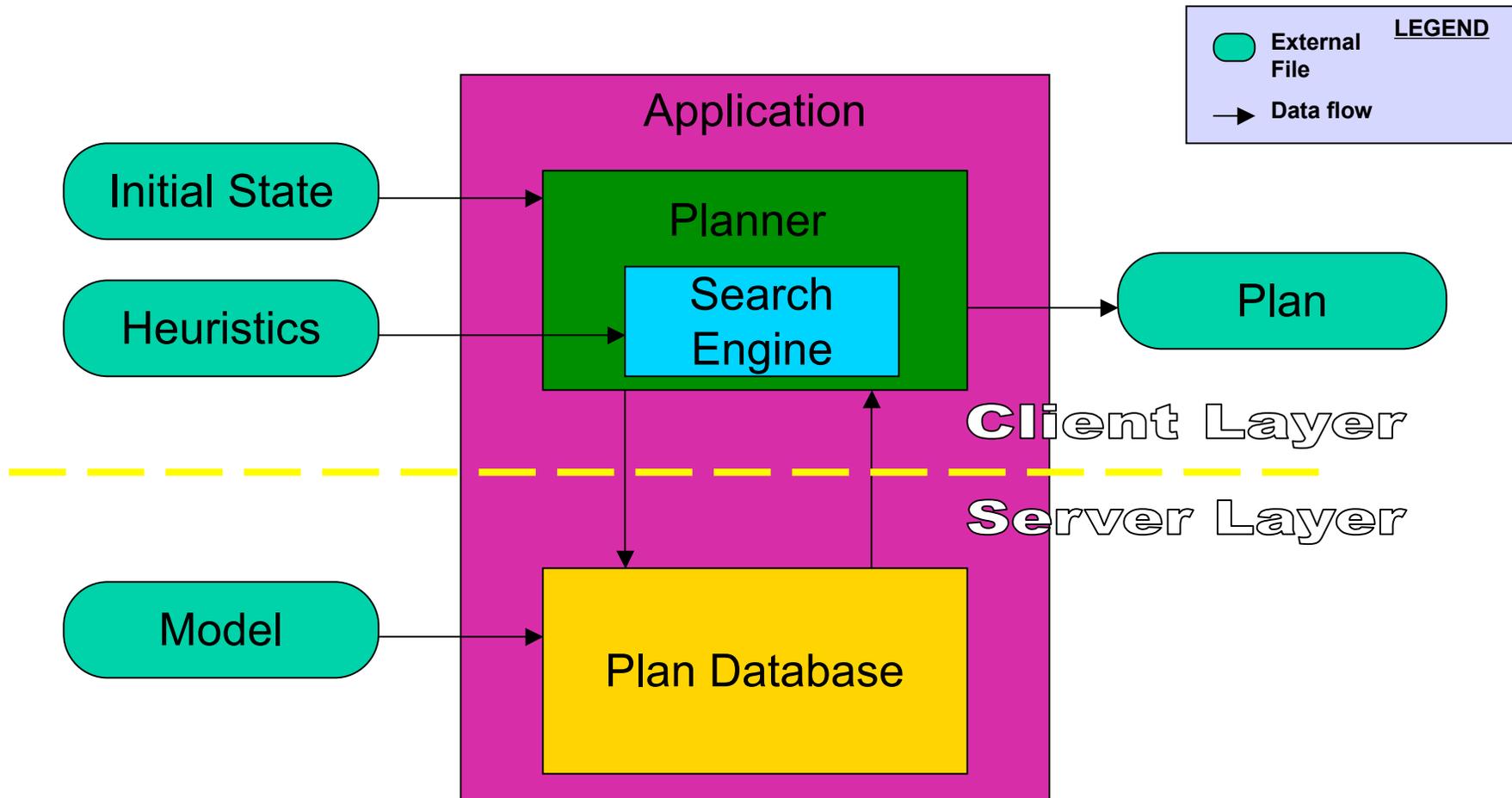
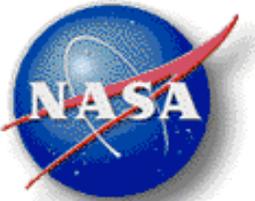
Mission-oriented research

- Earth-observing satellite scheduling project (EOS)
- SOFIA flight scheduling project (SOFIA)
- Contingent Planning for Mars rover operations
- Personal Satellite Assistant (PSA)
- Spoken Interface Prototype for PSA
- Space Interferometry testbed (SIM)

Research

- Intelligent Deployable Execution Agent (IDEA)
- LORAX Rover Power budgeting

Abstract Application Architecture



Plan Representation

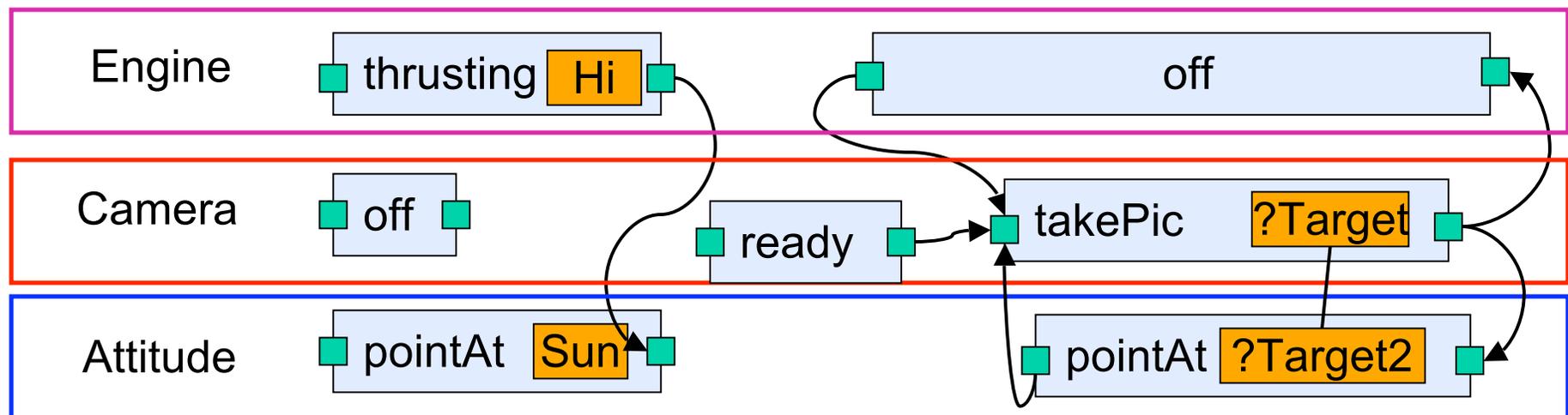
Flexible Time Intervals have Flexible Start, End and Duration

Parameterized Predicates describe actions and states

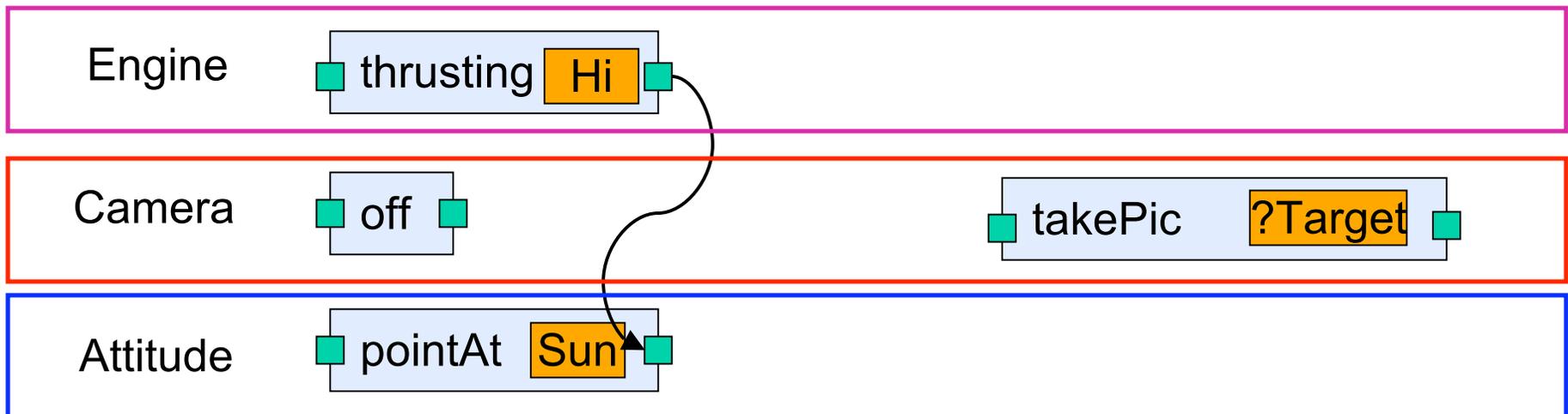
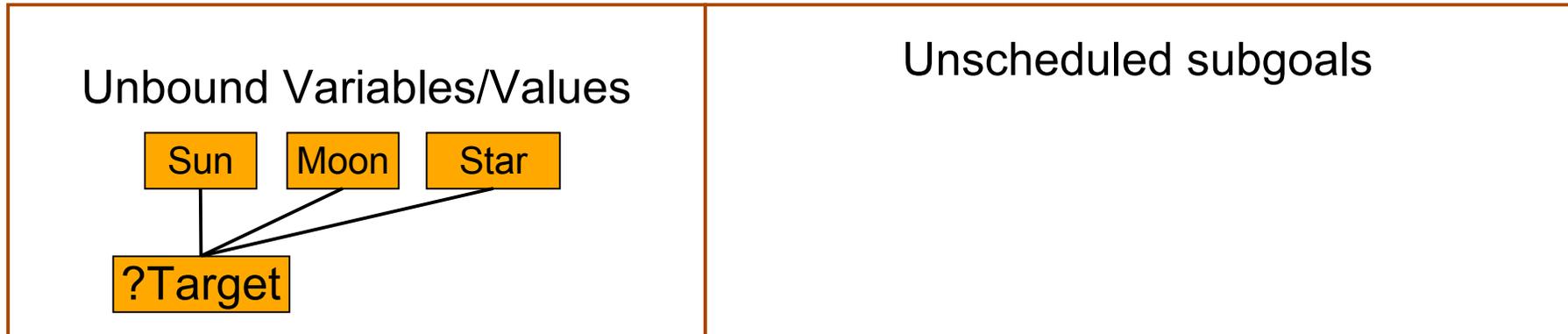
Token is a Parameterized Predicate over a Flexible Time Interval

Constraints defined between Tokens, Time Points, Parameters

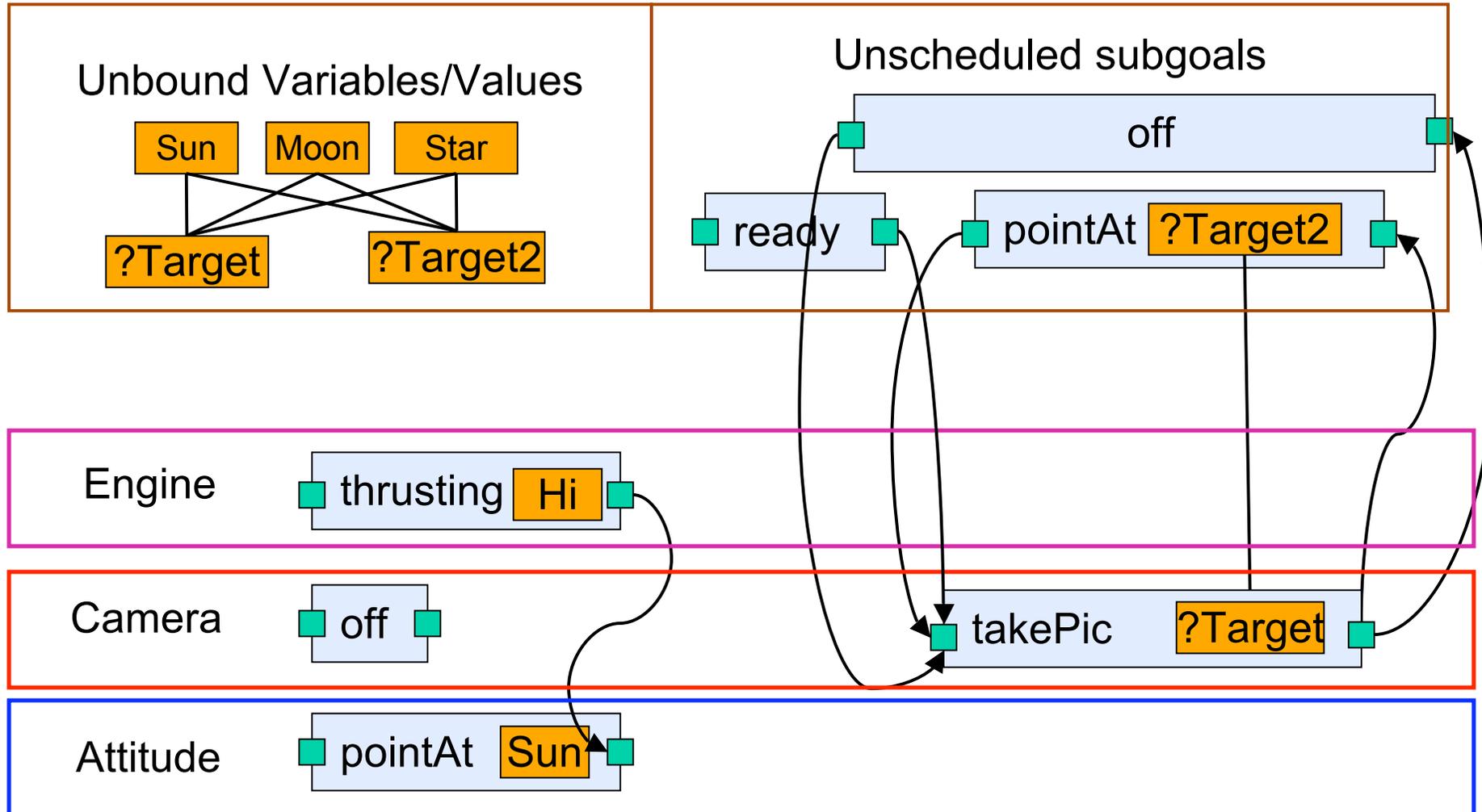
Timelines enforce temporal mutual exclusion over an object



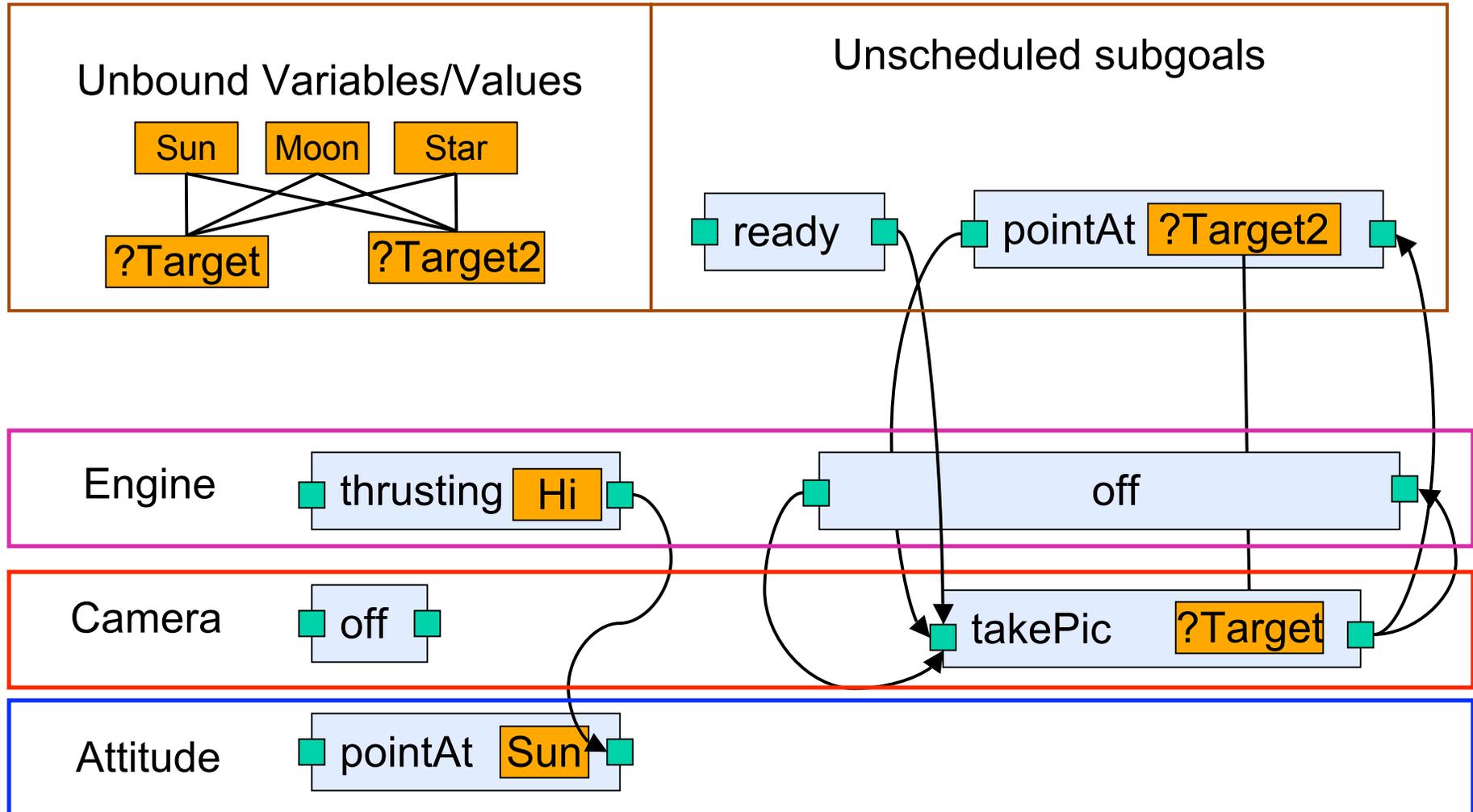
Insert takePic



Expand takePic subgoals



Insert off





Plan Database: how it works



A **Domain Model**:

- defines parts of the plan
- defines necessary relationships among them for valid plans

The **Plan Database** :

- maintains current plan
- maintains mapping between plan and constraint network
- supports plan modification and constraint inference

The **Planner**:

- checks status of current plan
- decides how to modify the plan



Sample Model



```
Camera::TakePic{  
  // Attitude must be constant throughout  
  contained_by(Attitude.pointAt at);  
  eq(at.location, rock);  
  
  // Engine must be off throughout  
  contained_by(Engine.off o);  
  
  // Preceded by readying operation  
  met_by(Ready r);  
  
  // Succeeded by stowing the instrument  
  meets(Stow c);  
}
```

Handling Heterogeneous Constraints



Distinguished constraint classes

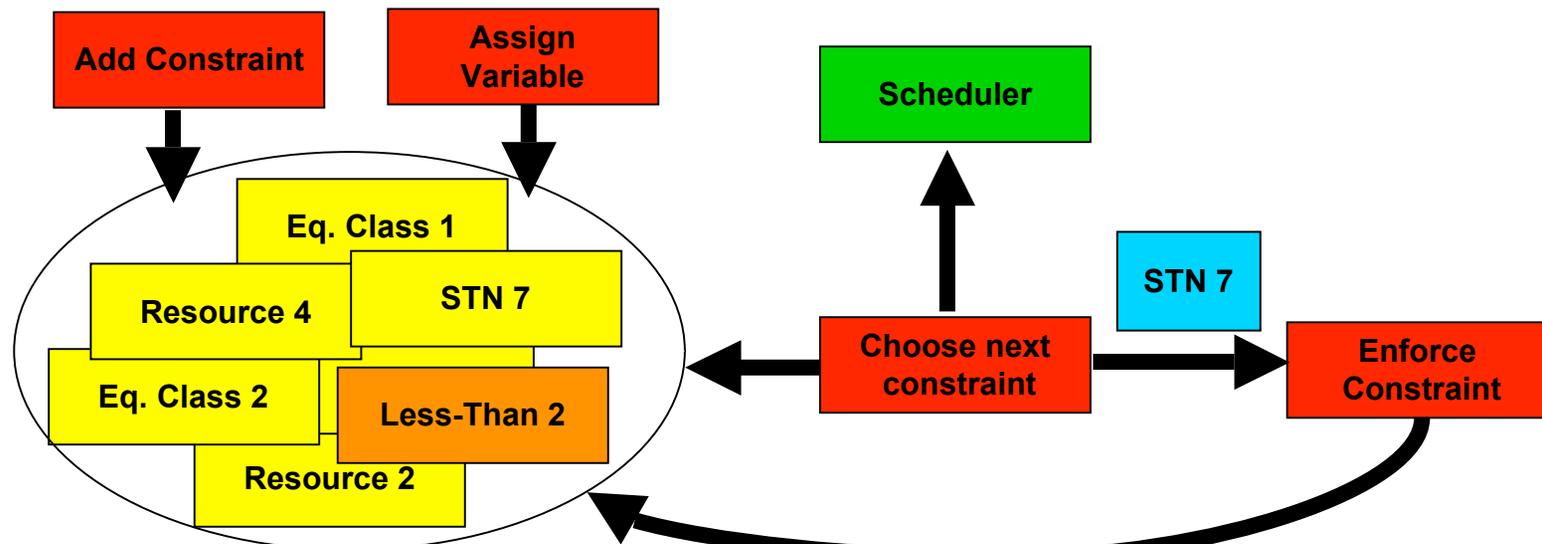
Generic, Temporal, Equivalence classes, Resources

Rules engine

Adds and removes constraints

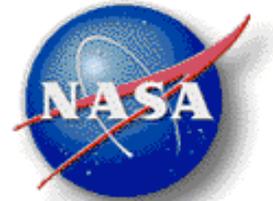
Triggering propagation via events

Scheduling of execution can be defined by user





PLASMA Customizability



Create new constraints, variables, tokens, objects

 Concept hierarchy supports customization (e.g. resource extends object)

Change rules governing legal plans

 e.g. Temporal flexibility vs timestamped sequences

Change order of domain rule enforcement

 e.g. Before or after tokens inserted on objects

Change scheduling constraint enforcement

 e.g. Resources before or after Temporal

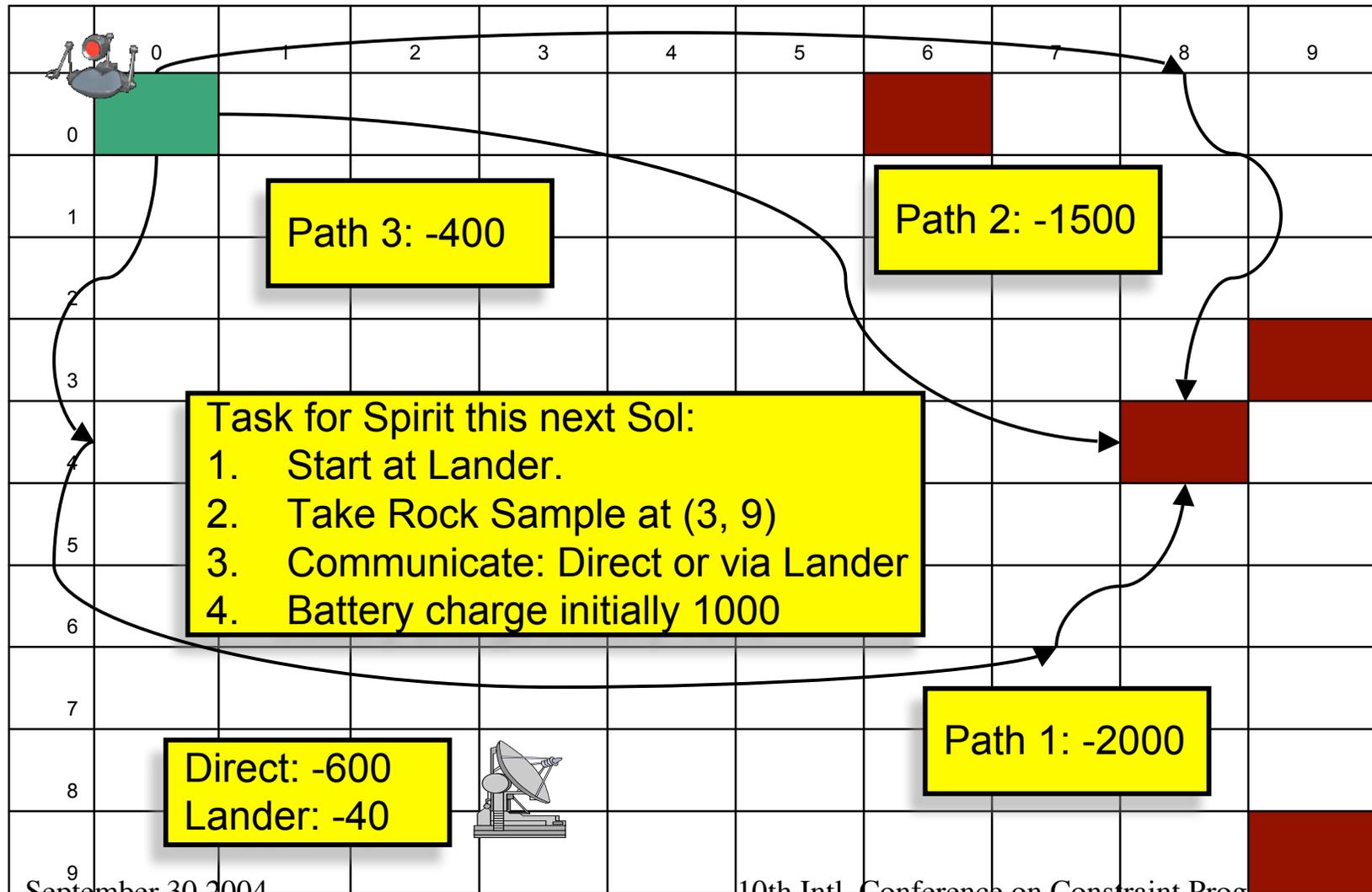
Customize planner search

 e.g. flaws and decision management

Change event model and event handling

Custom Logging

Example: Rover Rock Sampling



Objects with and without Tokens

Object

Rock

Member Variables (**Static** w.r.t. Time)

name(rock4)

x(3)

y(9)

Object

Navigator

Member Variables (**Variable** w.r.t. Time)

At(lander)

Going(lander, rock4)

At(rock4)

Object

Instrument

Member Variables (**Variable** w.r.t. Time)

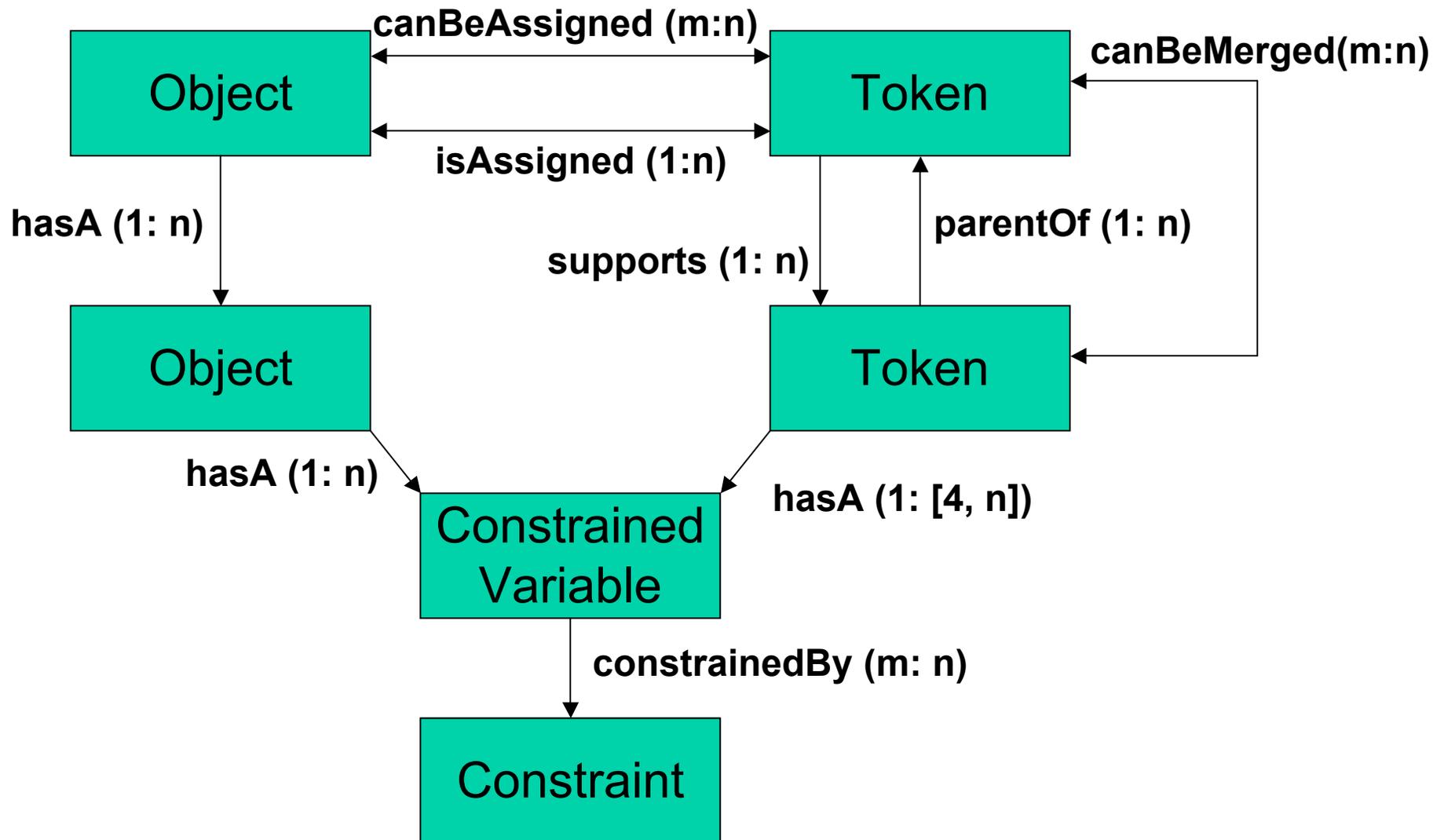
Stowed

Unstow

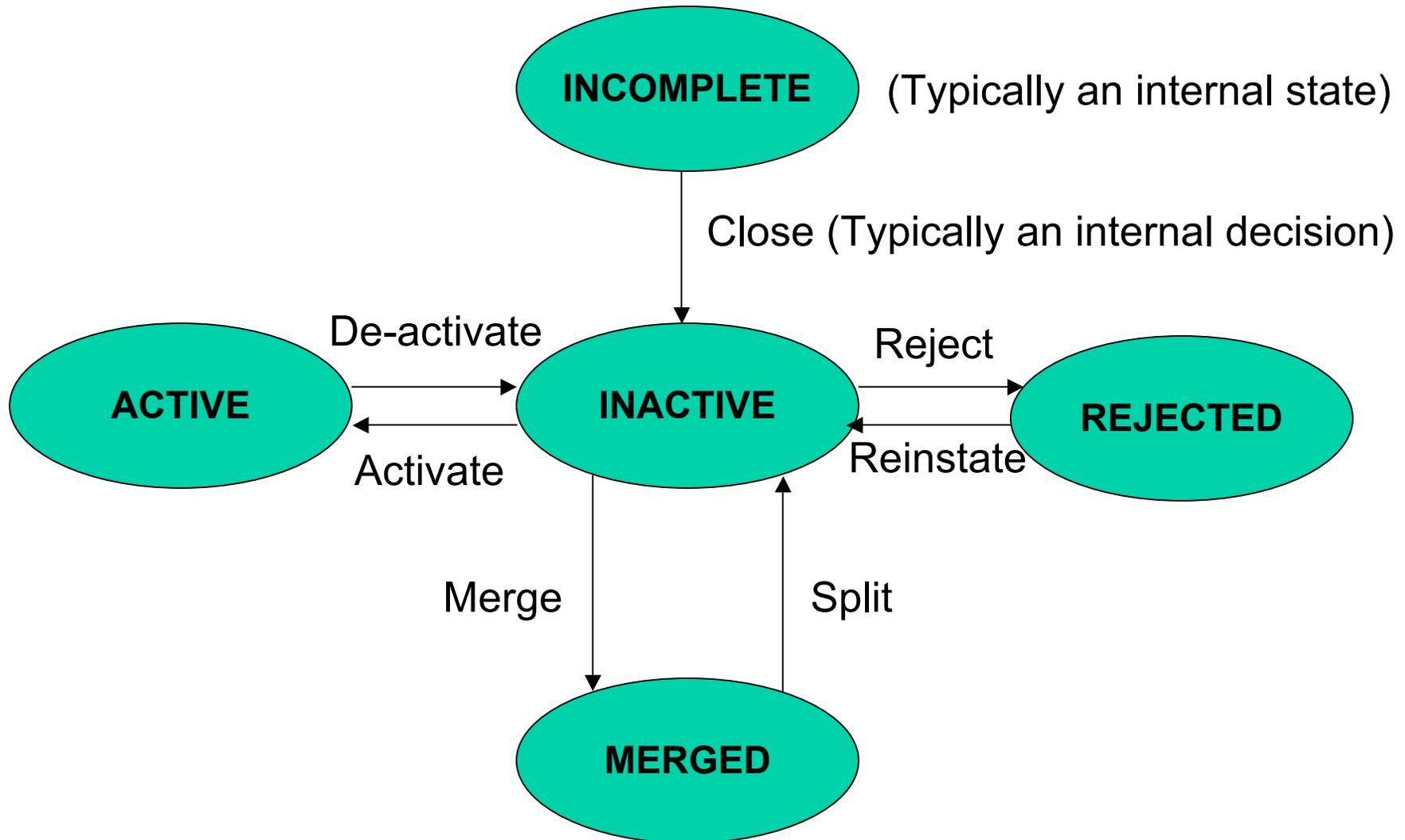
Place(rock4)

TakeSample(rock4)

Object - Token Relationships



Token State Transition Model





Flaw/Decision Model



Variable Decisions (resolve unbound variables):

- Specify (var, val) / Reset (var)

Token Decisions (resolve inactive tokens):

- Activate(Token t) / Deactivate(Token t)
- Merge (Token t1, Token t2) / Split(Token t1)
- Reject(Token t1) / Reinstate (Token t1)

Object Decisions (resolve when Object hasTokensToOrder):

- Constrain(Object o, Token t) / Free(Token t)
- Constrain(Object o, Token t1, Token t2) / Free(Token t1)

PLASMA Framework & Components

