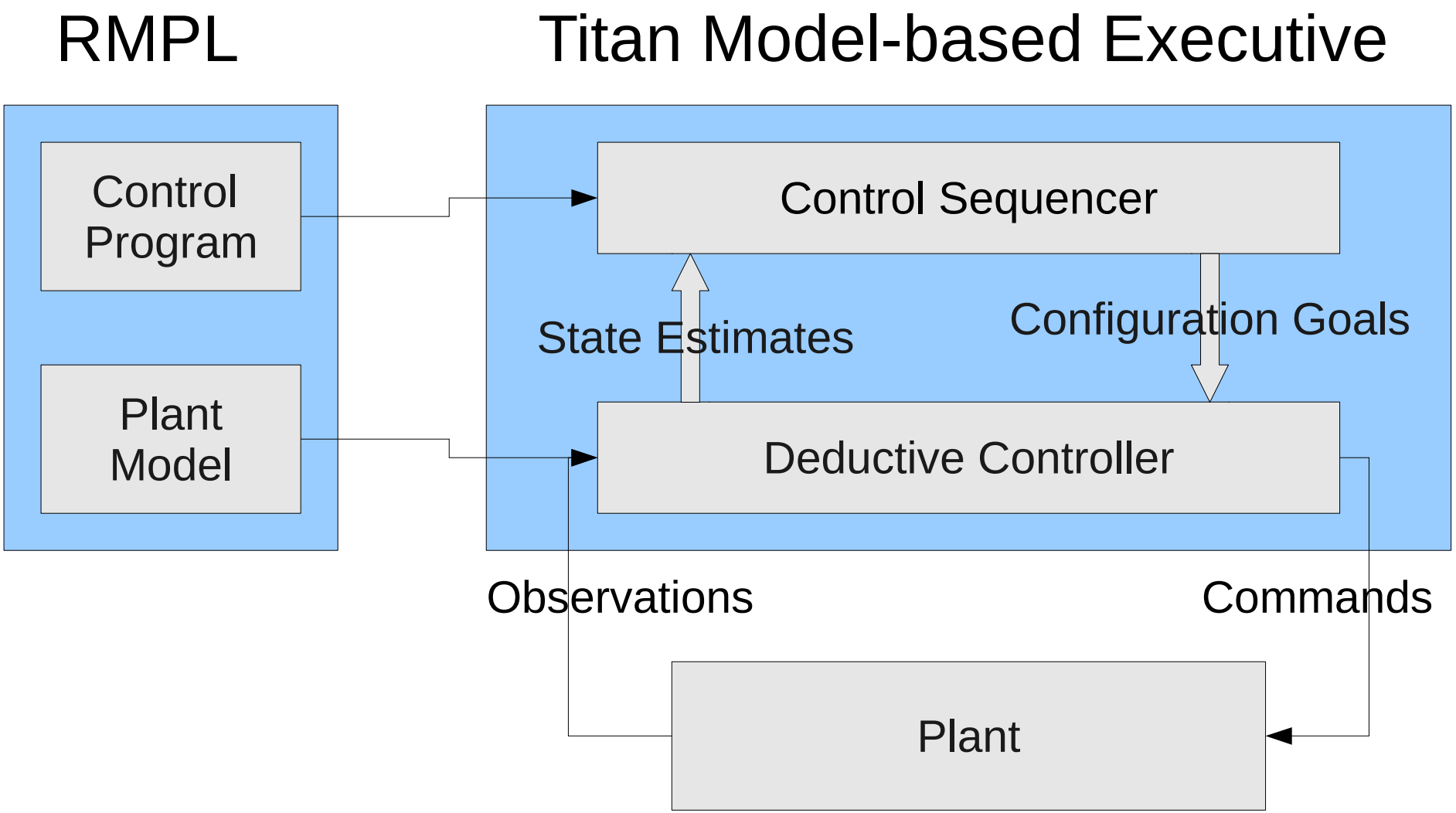
Titan

October 1st 2010

James Marshall

Overview


RMPL
Titan Model-based Executive
Control Program
Plant Model
Control Sequencer
State Estimates
Configuration Goals
Deductive Controller
Observations
Commands
Plant

Control Sequencer

- Not very interesting
- Interpreter for RPML code

Deep Space 1

- Experimental NASA satellite
- Remote Agent Experiment (RAX)
  - Precursor to Titan work
- Mostly successful:
  - Correctly generated a plan (although unexpected)
  - Handled simulated failures correctly
- Some problems

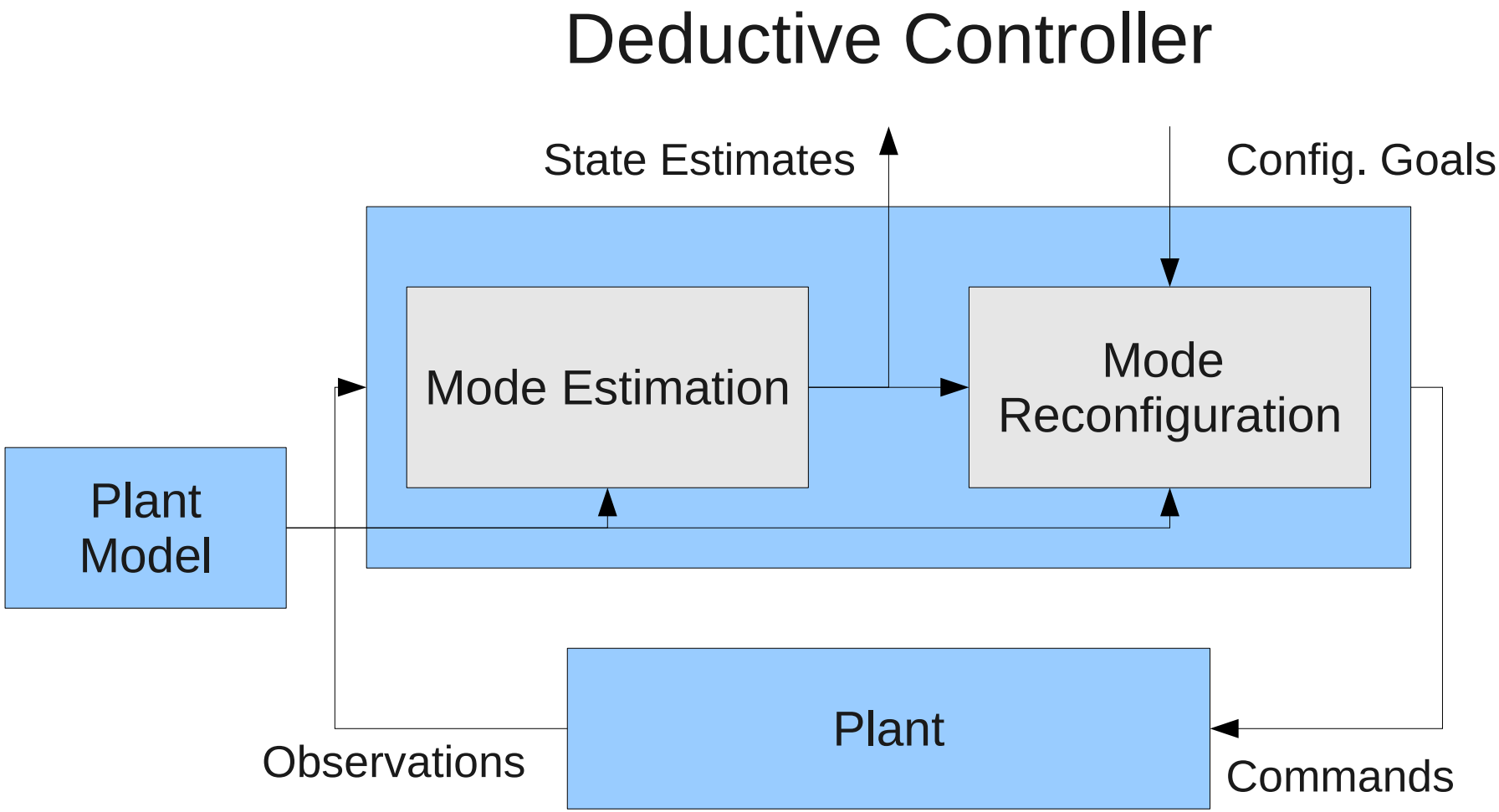
Lessons Learned

- Model Validation needed
  - Model – descriptions of hardware components
  - Validation of the system itself would be nice
- Unpredictable in novel situations
  - Search is complete but:
    - Not always completed within constraints
    - Can't test all possible scenarios

What Titan Adds

- Probabilistic
- Automatic failure recovery
- Fast?
- Modularity?

Probabilistic: Mode Estimation



Mode Estimation

- Input:
  - Observations
  - Plant Model (POMDP)
- Calculates probability of every state
  - Based off Hidden Markov Model belief state update
- Has “history”: probabilities conditioned by previous moves
- Output: Most likely current state

Failure Recovery: Mode Reconfiguration

- Input:
  - Most likely current state (ME)
  - Configuration goal (CS)
  - Plant Model (POMDP)
- Generates Control Action -
  - Moves system closer to Configuration Goal
  - Maximizes reward
  - Uses a nominal transition to new most likely state

Plant Model: POMDP... again

- Represents:
  - Hardware state
  - Observable variables (probabilistic relation to state)
  - Control variables
  - Transitions between states (probabilistic), including failures
  - Reward values for states

MR and POMDP

- That's how MR can select the “best” next command
  - Taking “future” into account
- Since nothing is certain, confirm and possibly re-evaluated after each command
- Errors are thus dealt with, without “re-planning”
  - Control Sequencer isn't needed to recover

Speed

- Just like Remote Agent, too slow
- State space grows exponentially
- Can't even limit possible states by observations:
  - Each component could be failing in a novel way

Implementation of POMDP

- Constraint Automata
  - One for each component
  - Operate synchronously
  - Mode variable
  - attribute variables with constraints (behavior)
  - Transition functions (with probability distribution)
  - Reward function

Concurrent Constraint Automata

- CAs are not enough
- Interconnections between CAs
- Interconnections with environment
- This is the Plant Model

Example: Valve

- Driver
  - Mode
    - On, Off, Resettable
  - On: (constraints)
    - cmd_in = cmd_out
  - Off:
    - cmd_out = none
  - Attr:
    - Control: cmd_in
    - Dependent: cmd_out
- Valve
  - Mode
    - Open, Closed, Stuck
  - Attr:
    - Dependent: vcmd
    - Observable: inflow, outflow
- Interconnections
  - cmd_out = vcmd

Valve Transitions

- Nominal, Current: driver On
  - cmd_in = off → Off
  - cmd_in != off → On
- Nominal, Current: driver Resettable
  - cmd_in = reset → On
  - cmd_in != reset → Resettable
- Failure, Current: driver On
  - True → Resettable

Open Sesame (err... Valve)

- Configuration Goal: (Valve = Open)
- Command: cmd_in = on
- Observe (inflow = 0, outflow = 0)
- Is the Driver failing or is the Valve stuck?

Belief Update

- How many belief states?
- New Observation!
  - Enumerate transitions
  - States that conflict: thrown out
  - States that predict: favor
- How do we enumerate transitions?

OpSat

- Well studied problem
- Control Variables, State constraints

Still too much

- Incremental Truth Maintenance discovers conflicts in candidates
- Conflicts used by conflict directed A* search to prune
- Now we have a (hopefully small) set of possible states

Mode Reconfiguration... again

- Take the most likely state from ME
- Only allow “reversible” transitions
  - Failures are the exception
- First, finds most desirable goal
  - Same process as ME
- Constant time planner maps out commands
  - Deferred to previous AI research

∃nd.