Timed Model-based Programming: Executable Specifications for Robust Critical Sequences

Michel D. Ingham
Brian C. Williams

Model-based Embedded Robotic Systems Group
MIT Space Systems Laboratory
MIT Artificial Intelligence Laboratory
June 10th, 2003

Motivation

Deep space exploration:
- highly uncertain environment
- require highly robust system

Mission-critical sequences:
- launch & deployment
- planetary fly-by
- orbital insertion
- entry, descent & landing

Problem Statement

- Traditional programming can lead to "brittle" sequences:
  - complexity of plant interactions
  - complexity of control specification
  - complexity of off-nominal behavior
- Time is central to the execution of mission-critical sequences:
  - plant spec: component behavior includes latency and evolution
  - control spec: hard-coded delays in sequence capture state knowledge
- Robust executive must consider time in its control and behavior models, in addition to reactively managing complexity

Current "State of the Practice"

Non-Critical Mission Sequences:
- Time-tagged nominal command sequences

```
GS,SITURN,490UA,BOTH,96-355/03:42:00.000;

CMD,7GYON,  490UA412A4A,BOTH,      96-355/03:47:00.000,  ON;
CMD,7MODE,  490UA412A4B,BOTH,      96-355/03:47:02:000,  INT;
CMD,6SVPN,  490UA412A6A,BOTH,      96-355/03:48:30:000,  2;
CMD,7ALRT,  490UA412A4C,BOTH,      96-355/03:50:32:000,  6;
CMD,7SAFE,  490UA412A4D,BOTH,      96-355/03:52:00:000,  UNSTOW;
CMD,6ASSAN, 490UA412A6B,BOTH,      96-355/03:56:08:000,  GV,153,IMM,231,
                                                         GV,153;
CMD,7VECT,  490UA412A4E,BOTH,      96-355/03:56:10.000,  0,191.5,6.5,
                                                         0.0,0.0,0.0,
                                                         96-350/
                                                         00:00:00.000,MVR;
SEB,SCTEST, 490UA412A23A,BOTH,     96-355/03:56:12.000,  SYS1,NPERR;
CMD,7TURN,  490UA412A4F,BOTH,      96-355/03:56:14.000,  1,MVR;
MISC,NOTE,  490UA412A99A,,         96-355/04:00:00.000,  ,START OF TURN;,
CMD,7STAR,  490UA412A406A4A,BOTH   96-355/04:00:02.000,  7,1701,
                                                         278.813999,38.74;
CMD,7STAR,  490UA412A406A4B,BOTH,  96-355/04:00:04.000,  8,350,120.455999,
                                                         -39.8612;
CMD,7STAR,  490UA412A406A4C,BOTH,  96-355/04:00:06.000,  9,875,114.162,
                                                         5.341;
CMD,7STAR,  490UA412A406A4D,BOTH,  96-355/04:00:08.000,  10,159,27.239,
                                                         89.028999;
CMD,7STAR,  490UA412A406A4E,BOTH,  96-355/04:00:10.000,  11,0,0.0,0.0;
CMD,7STAR,  490UA412A406A4F,BOTH,  96-355/04:00:12.000,  21,0,0.0,0.0;
```

Current "State of the Practice"

Non-Critical Mission Sequences:
- Time-tagged nominal command sequences
- If absolutely necessary, conditional behavior via rule-based monitors or hard-coded state machines

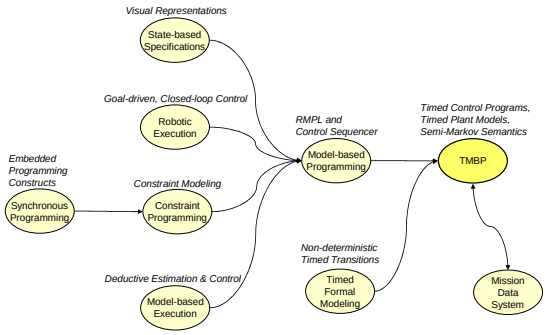
5.1.2.2.6	If the supply current for OXIDIZER TANK PRIMARY HEATER, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF OXIDIZER TANK PRIMARY HEATER.
5.1.2.2.7	If the supply current for HELIUM TANK PRIMARY HEATER, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF HELIUM TANK PRIMARY HEATER.
5.1.2.2.8	If the supply current for STAR TRACKER A, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF STAR TRACKER A.
5.1.2.2.9	If the supply current for STAR TRACKER B, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF STAR TRACKER B.
5.1.2.2.10	If the supply current for IMU PPSM-A, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF IMU PPSM-A.
5.1.2.2.11	If the supply current for IMU PPSM-B, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF IMU PPSM-B.
5.1.2.2.12	If the supply current for REACTION WHEEL 1, multiplied by the bus voltage, is greater than TBD watts, then the FPP shall issue a command to turn OFF REACTION WHEEL 1. In addition, the corresponding rules that monitor power dissipation of the remaining three reaction wheels shall be disabled. REACTION WHEEL 1 shall be flagged as "unavailable" to the G&C task in the MP.

Current "State of the Practice"

Non-Critical Mission Sequences:
- Time-tagged nominal command sequences
- If absolutely necessary, conditional behavior via rule-based monitors or hard-coded state machines
- Usual off-nominal behavior response is "safe mode":
  - costly ground ops
  - lost science opportunities

Critical Mission Sequences:
- Standard safing mechanism is disabled
- Hard-coded fault protection via highly-specialized s/w modules:
  - ad-hoc
  - complex
  - expensive to generate and test

Related Work



Related Work

- State-based Specifications
  - StateCharts (Harel, '87)
  - Timed StateCharts (Kesten & Pnueli, '92)
- Synchronous Programming
  - Esterel (Berry & Gonthier, '92)
  - Lustre (Halbwachs, '93)
- Constraint Programming
  - TCC (Saraswat, Jagadeesan & Gupta, '94)
- Robotic Execution
  - RAPs (Firby, '89)
  - ESL (Gat, '96)
  - TDL (Simmons, '98)
- Timed Formal Modeling
  - Timed Transition Systems (Henzinger, Manna, & Pnueli, '92)
  - Timed Automata (Alur & Dill, '94)
- Model-based Execution
  - GDE, Sherlock (deKleer & Williams, '87-'89)
  - Livingstone (Williams & Nayak, '96-'97)
  - Livingstone2 (Kurien & Nayak, '00)
- Model-based Programming
  - RBurton (Williams & Gupta, '99)
  - Titan (Williams, Ingham, Chung & Elliott, '03)
- Mission Data System
  - MDS (Dvorak, Rasmussen, et al., '00)

Principal Contributions

1. Language definition
   - Textual & graphical programming languages for control spec
   - Extension of plant modeling language to capture timed effects
2. Formal execution semantics
   - Plant modeled as factored Partially Observable Semi-Markov Decision Process (POSMDP)
   - Control program expressed as timed deterministic automaton
   - Execution defined in terms of legal plant state evolutions
3. Algorithm specification & implementation
   - Execution of timed control specifications
   - Reasoning on timed plant models (for estimation and reconfiguration)
4. Architecture design & implementation
   - Modular, state-based & fault-aware
   - Demonstrated on representative mission scenario

Objectives & Outline

Capability Overview
- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution

Technical Details
- Execution semantics
- Executive implementation
- Contributions and future directions

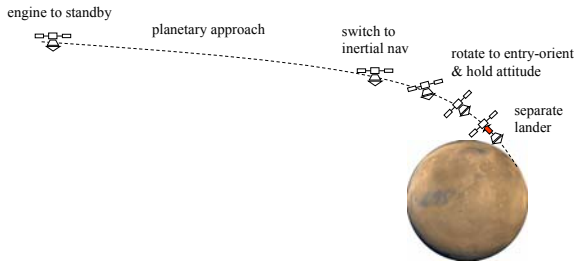
Objectives & Outline

- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

Objectives & Outline

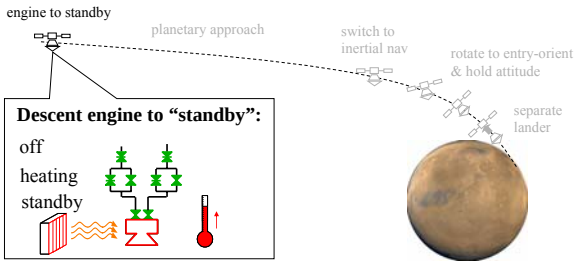
- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

Mars Entry Sequence: State-based Specification

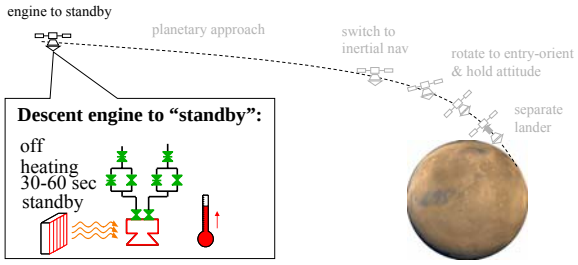


(Loosely based on Mars Polar Lander Entry Sequence)

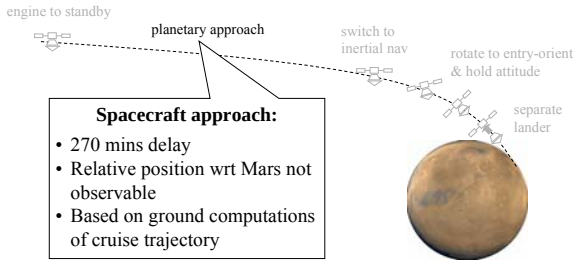
Mars Entry Sequence: State-based Specification



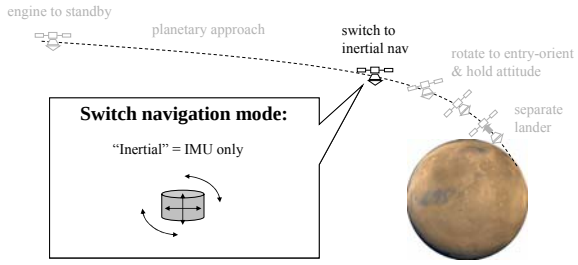
Mars Entry Sequence: State-based Specification



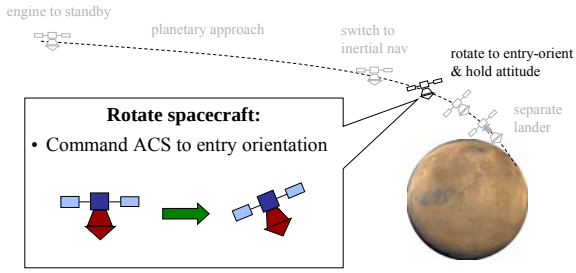
Mars Entry Sequence: State-based Specification



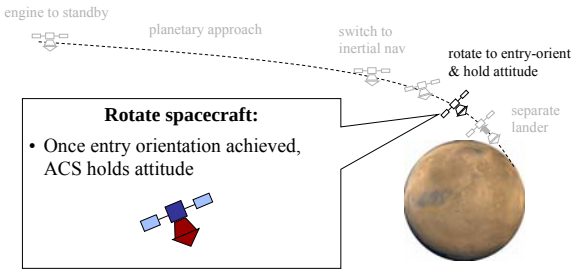
Mars Entry Sequence: State-based Specification



Mars Entry Sequence: State-based Specification



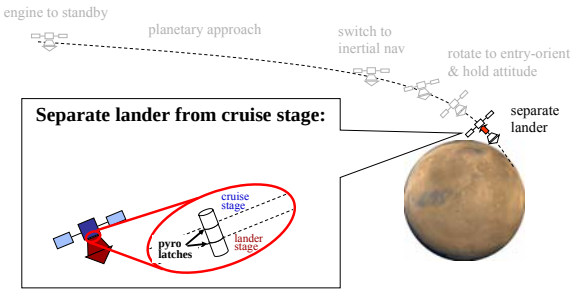
Mars Entry Sequence: State-based Specification



Rotate spacecraft:

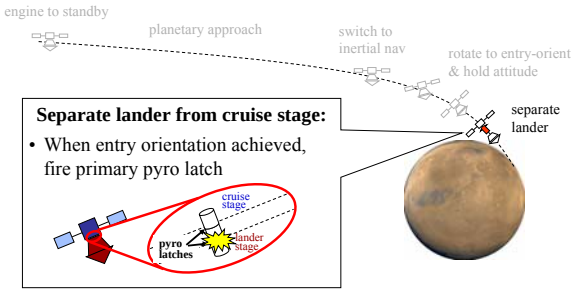
- Once entry orientation achieved, ACS holds attitude

Mars Entry Sequence: State-based Specification



Separate lander from cruise stage:

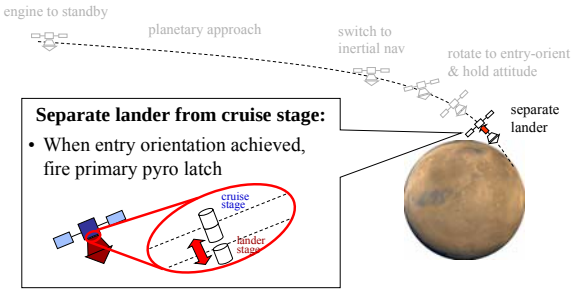
Mars Entry Sequence: State-based Specification



Separate lander from cruise stage:

- When entry orientation achieved, fire primary pyro latch

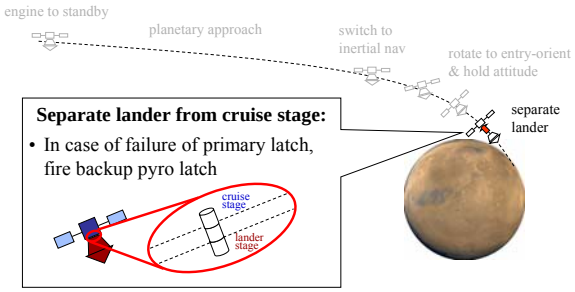
Mars Entry Sequence: State-based Specification



Separate lander from cruise stage:

- When entry orientation achieved, fire primary pyro latch

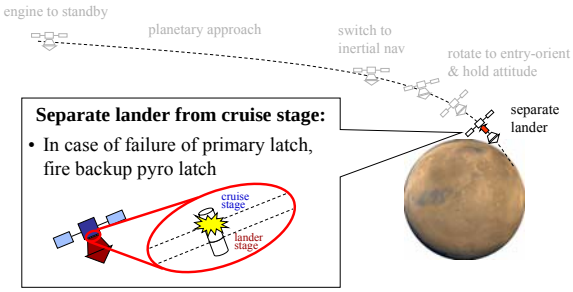
Mars Entry Sequence: State-based Specification



Separate lander from cruise stage:

- In case of failure of primary latch, fire backup pyro latch

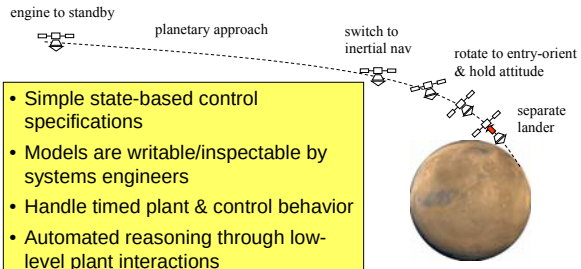
Mars Entry Sequence: State-based Specification



Separate lander from cruise stage:

- In case of failure of primary latch, fire backup pyro latch

Key Features of Executive



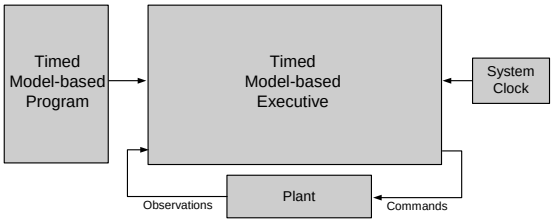
- Simple state-based control specifications
- Models are writable/inspectable by systems engineers
- Handle timed plant & control behavior
- Automated reasoning through low-level plant interactions
- Fault-aware (in-the-loop recoveries)

TMBP for Mars Science Lab

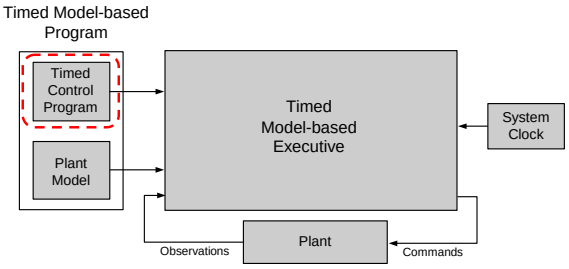
Objectives & Outline

- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

Timed Model-based Program

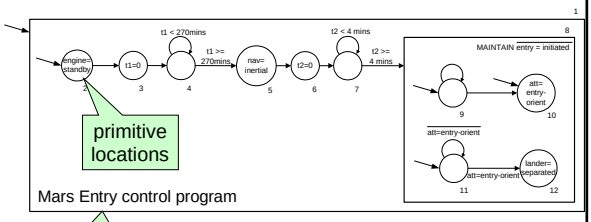


Timed Model-based Program



Timed Hierarchical Constraint Automata

- Graphical specification language for control programs, in spirit of Timed StateCharts
- Writable, inspectable by systems engineers

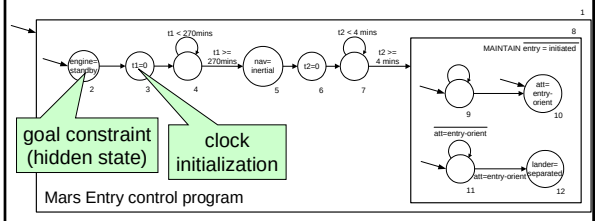


composite locations

➢ *compact encoding: multiple locations can be simultaneously marked*

Timed Hierarchical Constraint Automata

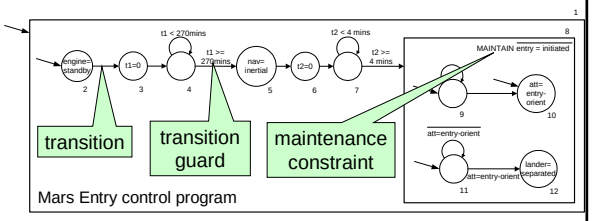
- Graphical specification language for control programs, in spirit of Timed StateCharts
- Writable, inspectable by systems engineers



- *act on hidden state*
- *clocks provide timing mechanism*

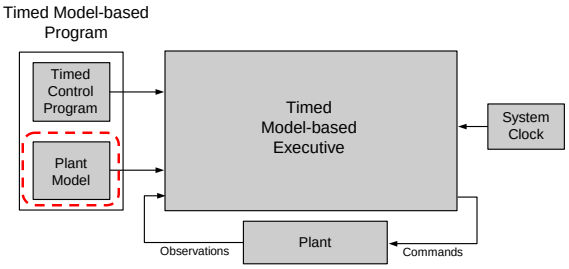
Timed Hierarchical Constraint Automata

- Graphical specification language for control programs, in spirit of Timed StateCharts
- Writable, inspectable by systems engineers



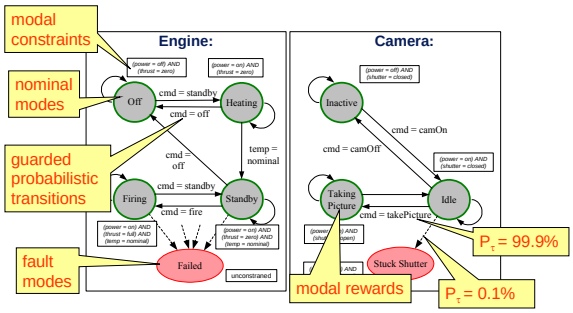
- *conditioned on time & state constraints*

Timed Model-based Program



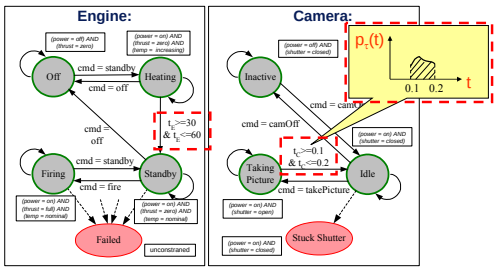
Concurrent Constraint Automata

- Variant of *Factored POMDP* (state not directly observable, next state depends on current state)



Timed Concurrent Constraint Automata

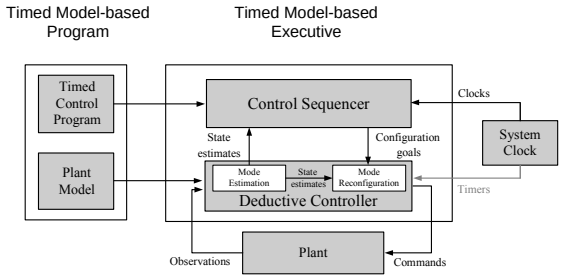
- Variant of *Factored POSMDP* (state not directly observable, next state depends on current state *& time spent in state*)
- Extend Concurrent Constraint Automata to timed behavior



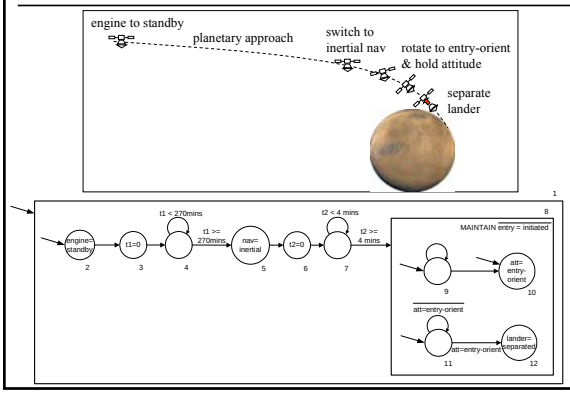
Objectives & Outline

- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

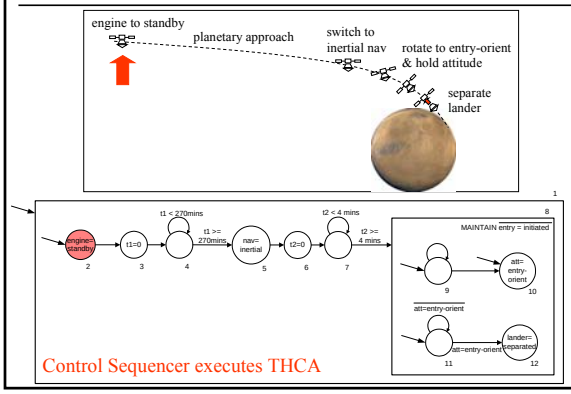
Timed Model-based Executive Architecture



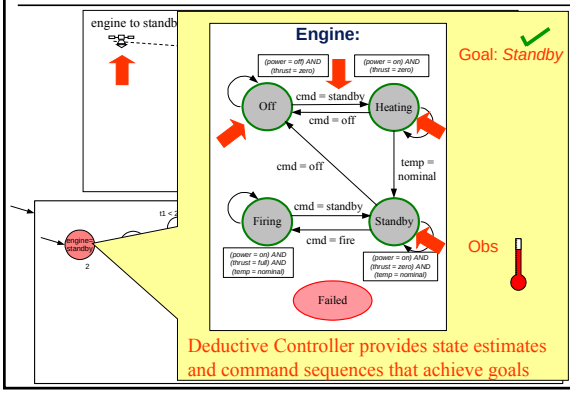
Mars Entry Example



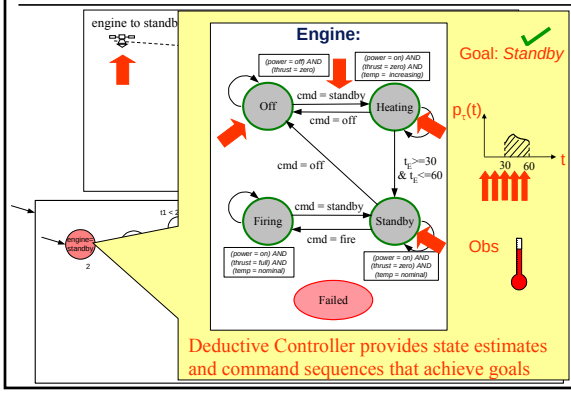
Mars Entry Example



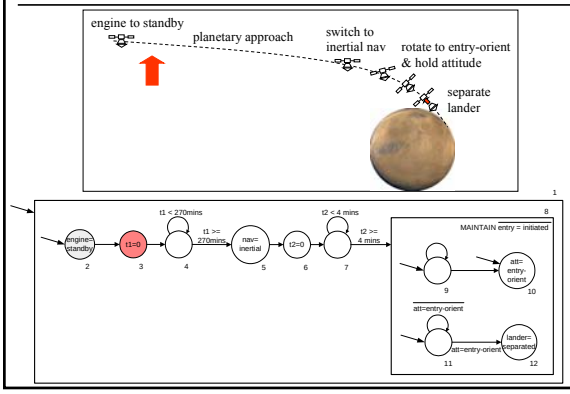
Mars Entry Example



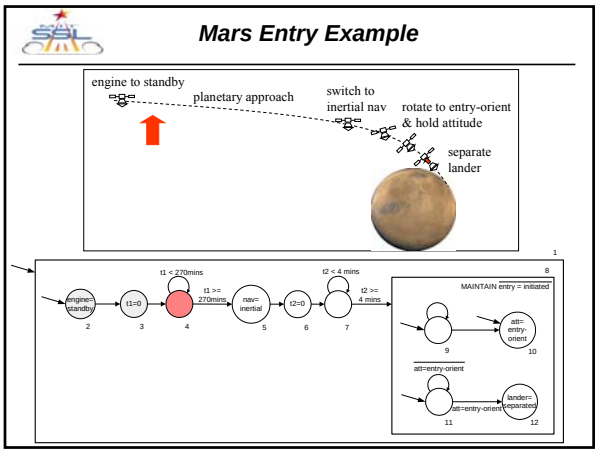
Mars Entry Example



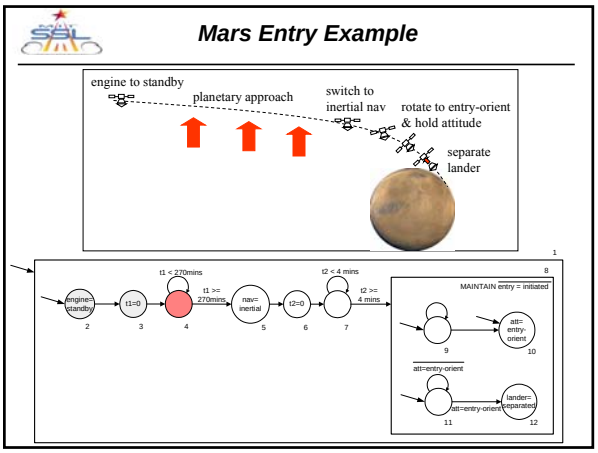
Mars Entry Example



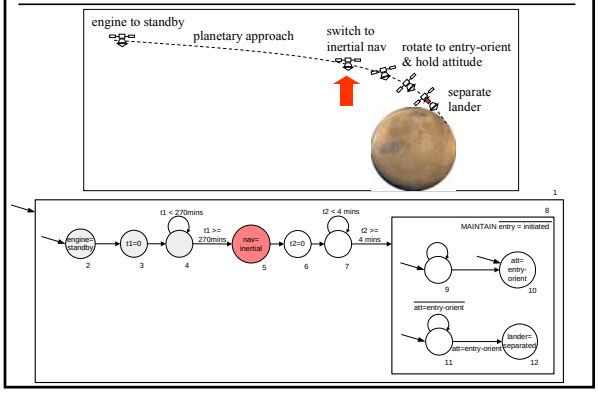
Mars Entry Example



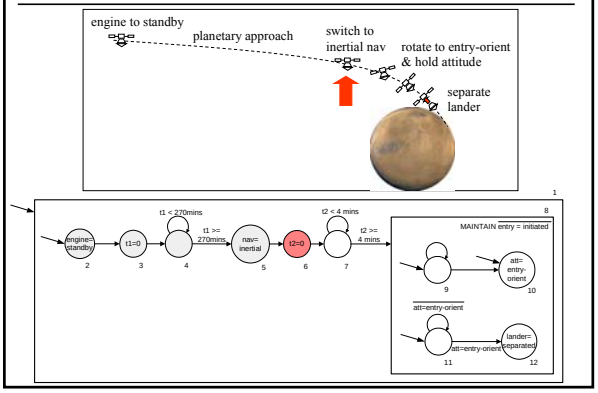
Mars Entry Example



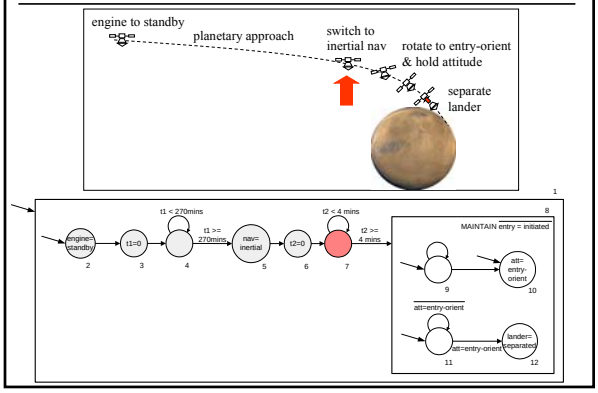
Mars Entry Example



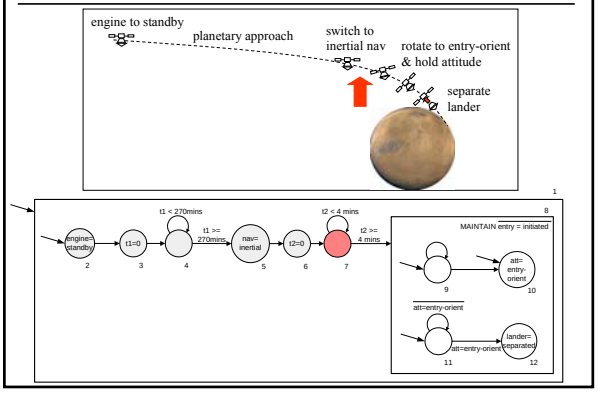
Mars Entry Example

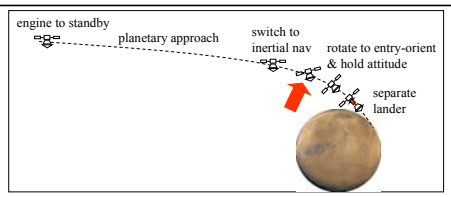


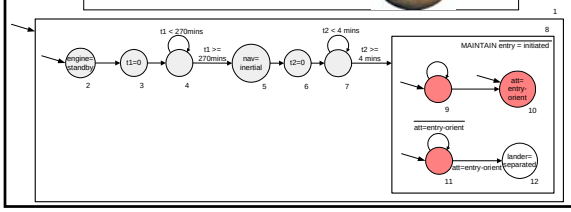
Mars Entry Example

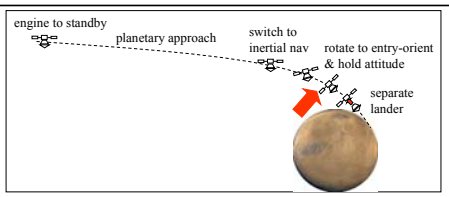


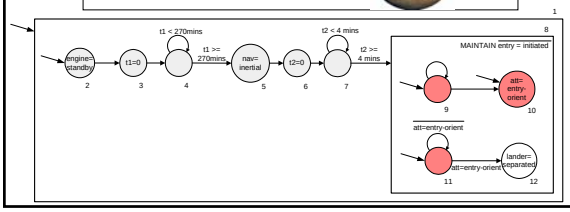
Mars Entry Example

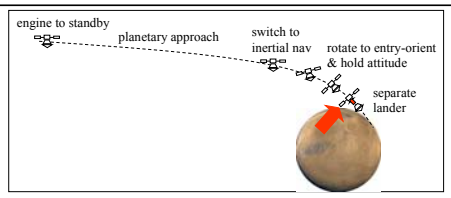


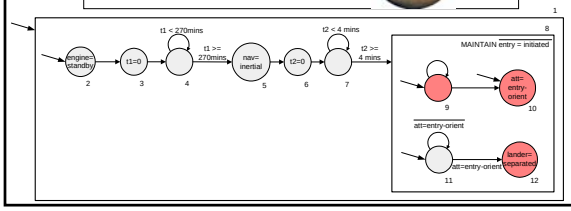


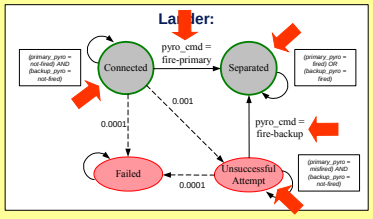


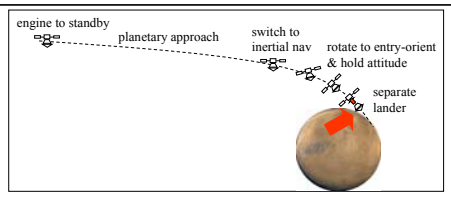


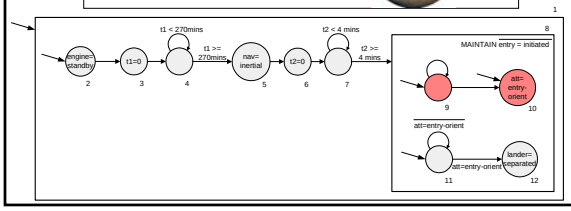


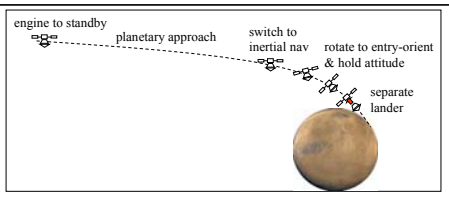


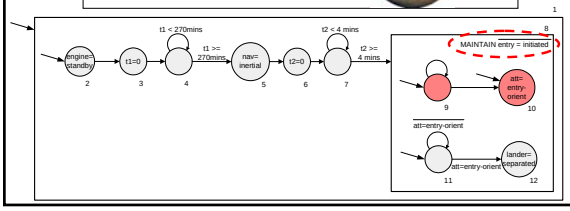












Mars Entry Example

Mars Entry Example

Mars Entry Example

Mars Entry Example

Goal: *Separated* ✓

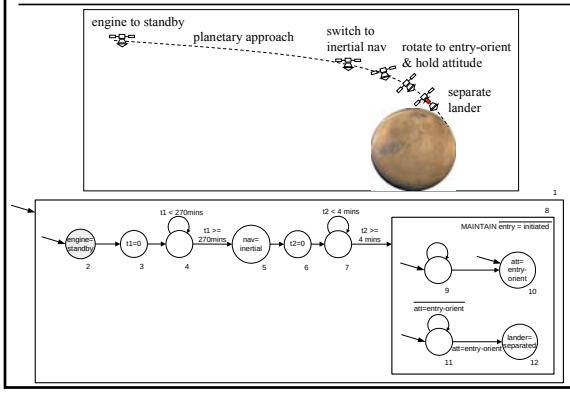
Obs: *backup pyro fired*

Model-based executive provides robustness in the goal-driven control loop

Mars Entry Example

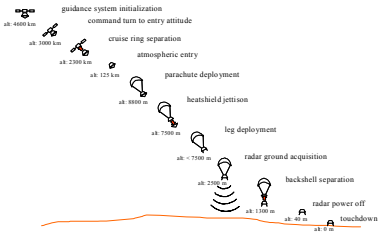
Mars Entry Example

Mars Entry Example



Complete EDL Scenario

- Proof-of-concept on a representative mission scenario: "Full" Entry, Descent and Landing scenario
- Control program (57 locations, 16 state vars, 6 clock vars)
- Plant model (~25 components, avg. 3-4 modes per component)



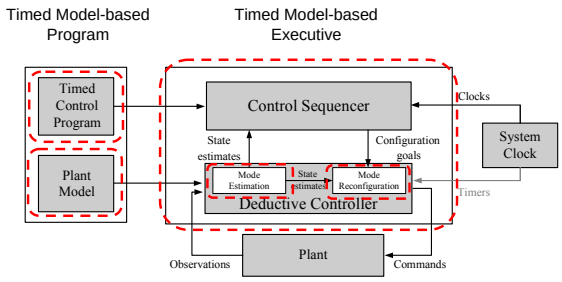
EDL Scenario Highlights Key Capabilities

- Nominal operations:
  - Execution conditioned on state constraints
  - Execution conditioned on time constraints
  - Nominal mode tracking through commanded and timed transitions
  - Accept configuration goal and generate appropriate command sequence (single-step, multi-step reconfigurations)
- Operations in the presence of faults:
  - Fault diagnosis through commanded transitions
  - Fault diagnosis through timed transitions
  - Recovery by repair (deductive controller)
  - Recovery by leveraging physical/functional redundancy (control sequencer, deductive controller)

Objectives & Outline

- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

TMBP Semantics



Plant Model

- Variables:

$$\Pi = \{\Pi^s, \Pi^c, \Pi^o\}$$

state vars, control vars, obs vars

Σ: full assignments σ over all vars in Π
Σ_s: plant states s
Σ_c: control actions μ
Σ_o: observations o

- Factored POMDP:

$$PM = \langle \Sigma, T, P_\Theta, P_T, P_O, R \rangle$$

transitions $\tau : \Sigma \rightarrow \Sigma_s$

initial state prob $P_\Theta(s_0)$

transition prob $P_T(s' \mid s, \mu)$

obs prob $P_O(o \mid s)$

state reward $R(s)$

Timed Plant Model

- Variables:

$$\Pi = \{\Pi^s, \Pi^c, \Pi^o, \Pi^t\}$$

timer vars

- Factored POSMDP:

$$TPM = \langle \Sigma, \mathrm{T}, P_\Theta, P_\mathrm{T}, P_\mathrm{O}, R \rangle$$

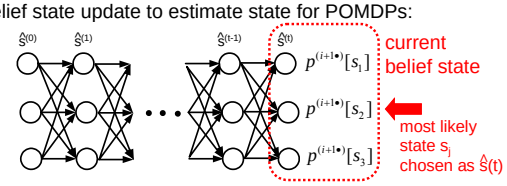
add Σ_t, set of assignments ν over all plant timers in Π^t

transitions $\tau : \Sigma \to \Sigma_s \times \Sigma_t$
- *conditioned on* ν
- *upon transition, subset of timers are reset*

transition prob $P_\mathrm{T}(s' \mid s, \mu, \nu)$

Mode Estimation

- *Given latest commands and observations, what is the most likely current state?*
- Belief state update to estimate state for POMDPs:



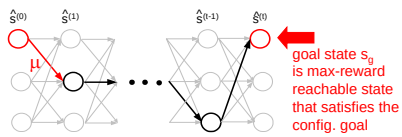
$$p^{(\bullet i+1)}[s_l] = \sum_{k=1}^{n} p^{(i\bullet)}[s_k] \mathrm{P}_\mathrm{T}(s_l \mid s_k, \mu)$$

$$p^{(i+1\bullet)}[s_l] = p^{(\bullet i+1)}[s_l] \frac{\mathrm{P}_\mathrm{O}(o^{(i+1)} \mid s_l)}{\sum_{k=1}^{n} p^{(\bullet i+1)}[s_k] \mathrm{P}_\mathrm{O}(o^{(i+1)} \mid s_k)}$$

Mode Reconfiguration

- *Given current belief state and configuration goal, what is the first control action from a policy that maximizes expected reward?*

Optimal policy π^*:



- Solve Bellman equation to compute optimal policy for POMDPs:

$$V^*(s) = \max_{\pi} E\left[\sum_{i=1}^{\infty} \gamma^i r_i\right]$$

$$\pi^*(s) = \arg\max_{\mu} \left[R_g(s) + \gamma \sum_{s' \in S} \mathrm{P}_\mathrm{T}(s' \mid s, \mu)\, V^*(s') \right]$$

Timed ME & MR

- **Problem**:
  For factored POSMDP, next state depends on current state, current control actions AND current <u>timer values</u>
- **Key Insight**:
  Define *"system state" = plant states ∪ plant timers*
- Timed ME can now use same belief state update equations, where *s* is now the *system state*
- Timed MR finds optimal policy based on *system state*, defines "wait" actions to accommodate non-deterministic timed transitions

Timed Control Program

- Control program:
  - program locations: L_{cp}
  - clocks: Π^t_{cp}
  - *deterministic* automaton:

$$TCP = \langle L_{cp}, \lambda_{cp}, \tau_{cp}, g_{cp}, \iota_{cp}, \Sigma_s, \Omega_{cp} \rangle$$

assignments ω to all clocks in Π^t_{cp}

initial program location

transitions between locations, conditioned on state & current clock values

config. goal $g_{cp}(l) \subset \Sigma_s$

clock init. $\iota_{cp}(l) \subset \Pi^t_{cp}$

Executive Semantics

- Interleaving model of execution
  cycle = discrete event + continuous phase
- Legal execution of TMBP:

Cycle start time	t_0	t_1	…
Plant state	$\hat{s}_0$		
Pgm location	l_0		
Pgm clocks	ω_0		

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal
4. next clock values are legal

$P_\Theta(\hat{s}_0) > 0$
$l_0 = \lambda_{cp}$
$\omega_0(x^t) = 0$, for all clocks x^t

Executive Semantics

- Interleaving model of execution
  cycle = discrete event + continuous phase
- Legal execution of TMBP:

Cycle start time	t_0	t_1	…
Plant state	$\hat{s}_0 \to \hat{s}_1$		
Pgm location	l_0		
Pgm clocks	ω_0		

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal
4. next clock values are legal

$$g_0 = g_{cp}(l_0)$$
$$\mu_0 = MR(PM, \hat{s}_0, g_0)$$
$$\hat{s}_1 = ME(PM, \hat{s}_0, \mu_0, \omega_0, o_1)$$

Executive Semantics

- Interleaving model of execution
  cycle = discrete event + continuous phase
- Legal execution of TMBP:

Cycle start time	t_0	t_1	…
Plant state	$\hat{s}_0 \to \hat{s}_1$		
Pgm location	$l_0 \to l_1$		
Pgm clocks	ω_0		

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal
4. next clock values are legal

$$l_1 = \tau_{cp}(l_0, \hat{s}_0, \omega_0)$$

Executive Semantics

- Interleaving model of execution
  cycle = discrete event + continuous phase
- Legal execution of TMBP:

Cycle start time	t_0	t_1	…
Plant state	$\hat{s}_0 \to \hat{s}_1$	$\hat{s}_1$	
Pgm location	$l_0 \to l_1$	l_1	
Pgm clocks	ω_0	ω_1	

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal
4. next clock values are legal

$$\omega_1 = \omega_0 + \Delta t,$$
$$\text{where } \Delta t = t_{ME} + t_{MR} + t_{CS}$$

Executive Semantics

- Interleaving model of execution
  cycle = discrete event + continuous phase
- Legal execution of TMBP:

Cycle start time	t_0	t_1	…
Plant state	$\hat{s}_0 \to \hat{s}_1$	$\hat{s}_1 \to \hat{s}_2$	…
Pgm location	$l_0 \to l_1$	$l_1 \to l_2$	…
Pgm clocks	ω_0	ω_1	…

Such that:
1. initial conditions are valid
2. next state is legal
3. next program location is legal
4. next clock values are legal

Implementation Approximations

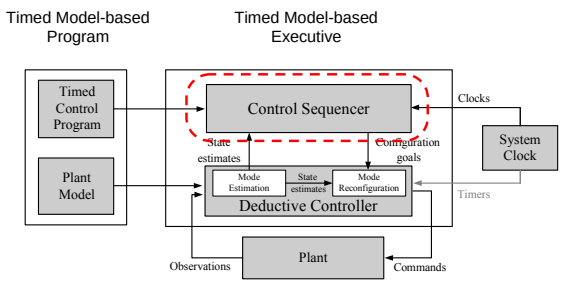
Mode Estimation:
- Full belief state update is computationally infeasible
- Assume probability of a few most-likely states dominates probability of other possible states
- Track a limited set of most-likely states, from one cycle to the next

Mode Reconfiguration:
- Assume probability of nominal behavior dominates off-nominal
- Assume reward of being in goal state dominates reward of getting to goal state
- Perform MR in 2 steps:
  - ***Goal Interpretation***: find the *max-reward goal state*, reachable via *nominal transitions*, that satisfies the configuration goal
  - ***Reactive Planning***: returns series of control actions that achieve the goal state

Objectives & Outline

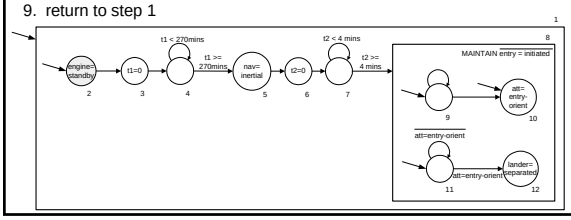
- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming:
  a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

Control Sequencer Implementation

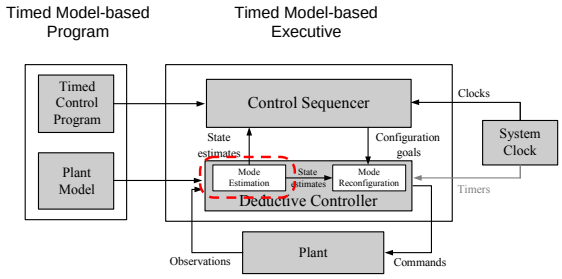


THCA Execution Algorithm

1. update active clocks
2. check maintenance constraints ← reactive preemption
3. assert clock initializations & state goals } ← goal-driven execution
4. request MR to take action
5. obtain new state estimate from ME } ← closed-loop execution
6. await incomplete goals
7. take enabled transitions ← progress due to goal achievement or preemption
8. mark new set of locations
9. return to step 1

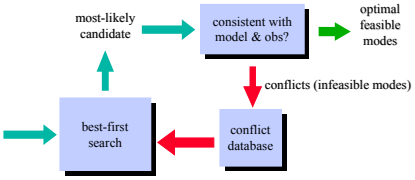


Deductive Controller Implementation



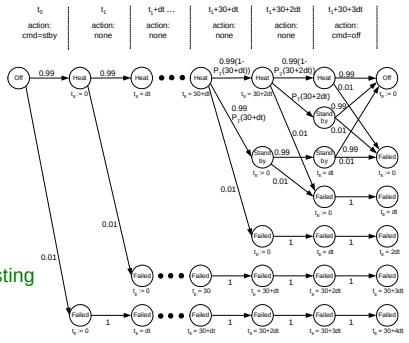
Mode Estimation

- Mode Estimation tracks a limited set of most-likely states
- Explores state space in best-first order:
  - Formulate Optimal Constraint Satisfaction Problem (OCSP), to identify "k-best" extensions to current trajectories ("shortest path" from set of current possible states to next possible states)
  - Solve using OPSAT engine:



Timed Mode Estimation

- For physical plants modeled as TCCA (POSMDP):



Bad news:
state space gets much larger…

Good news:
can leverage existing OPSAT engine!

TCCA Mode Estimation Algorithm (k = 1)

Given current system state $\mathbf{s}^{(i)}$, control action $\mu^{(i)}$, observation $o^{(i+1)}$ & current time t^{abs}:

1. Update timer values for $\mathbf{s}^{(i)}$
2. Compute probability associated with each possible next system state
3. Choose highest-probability system state
4. In this system state, reinitialize to zero any timers associated with components with changed modes
5. Return resulting system state

Perform steps 2 & 3 in best-first order, by framing as an Optimal Constraint Satisfaction Problem, then solving using OPSAT

TCCA Mode Estimation as OCSP

- Setup OCSP < **x**, f, C >:
  - *decision vars* **x**, such that dom[$\mathbf{x}_j$] = reachable target modes
  - *objective function* f(**x**) = prior probability of state **x**, i.e.:

$$\prod_j P_{\mathrm{T}j}(\mathrm{x}_j \mid s^{(i)}, \mu^{(i)}, t_j)$$

  - *constraint* C(**x**), such that $\mathrm{x} \wedge C_{M\mathrm{x}} \wedge o^{(i+1)}$ is consistent
- Solve using OPSAT

Objectives & Outline

- Timed Model-based Execution "in a nutshell"
- Timed Model-based Programming: a visual programming paradigm
- Illustration of Timed Model-based Execution
- Execution semantics
- Executive implementation
- Conclusions

Conclusions

- TMBP paradigm for visual programming of embedded systems:
  - THCA unify features of StateCharts, synchronous programming, constraint programming, timed automata and robotic execution languages
  - CCA allow constraint-based, probabilistic modeling of physical plants
  - (TCCA extend CCA to capture timed effects)
- Semantic specification for TMBP:
  - Physical plants modeled as factored PO(S)MDPs
  - ME as belief state update, MR as decision theoretic planning
  - Control programs modeled as deterministic automata
  - Control Sequencer steps control program location based on state & time
  - Execution of TMBP defined in terms of legal plant state evolutions
- Design & implementation of Timed Model-based Executive:
  - Execution architecture is modular, state-based & fault-aware
  - Control Sequencer executes THCA
  - ME performs approximate belief state update for (T)CCA
  - MR performs reactive planning for (T)CCA

Directions for Future Work

Theory:

- Formal verification (model checking) for timed plant models, timed control programs
- Extension to Hybrid Model-based Programming
  - Control programs can specify trajectories in terms of continuous and/or discrete states
  - Fold continuous estimators & controllers into Deductive Controller

Implementation:

- Improve Timed ME
  - Move costly M-B deduction offline, through compilation of the timed models
- Improve Timed MR
  - Consider time to reach goal to be included in cost

Backup Slides

Mars EDL: RMPL Code Excerpt

```
MarsEDL():: {
  do {
    EntrySequence(),
    DescentLandingSequence()
  } watching (landing = success)
}


EntrySequence():: {
  engine = standby;
  t1 = 0;
  when (t1 >= 16200.0) donext {
    nav = inertial;
    t2 = 0;
    when (t2 >= 240.0) donext {
      do {
        always (att = entry-orient),
        when (att = entry-orient) donext (lander = separated)
      } watching (entry = initiated)
    }
  }
}
```

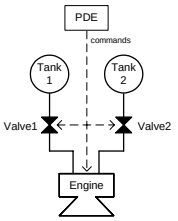
TCCA Mode Reconfiguration Algorithm Extensions

- Untimed MR algorithms have been extended to address:
  - timed transitions
  - irreversible actions

Desired MR behavior, given config. goal *Engine = Firing…*

RP should return the following control sequence:

{PDE-cmd = Turn-on,
Engine-cmd = Standby,
Wait until Standby mode is achieved,
Valve1-cmd = Turn-on,
Valve2-cmd = Turn-on,
Engine-cmd = Fire,
PDE-cmd = Turn-Off}



Assumptions/Limitations

1. Executive is "fast enough" to keep up with plant evolution
   - Mode of a component cannot change more than once per execution cycle, for ME algorithm to function correctly.
   - From Control Seq's perspective, transitions assumed to occur at execution cycle start times, and plant state is assumed to hold constant through to the time of the next execution cycle.
   - Duration of execution cycle is dictated by Ded. Contr. computation time, so require this computation time to be short.
   - This assumption limits effective resolution of time constraints in control programs and plant models.
2. Observations are provided to executive in a timely manner
   - In the absence of observations to refute nominal behavior, current exec implementation assumes nominal behavior.
   - In case of timed transitions, executive will take transition at "expected" nominal transition time (mean of transition PDF).
   - Observation associated with a transition should be received within the execution cycle that the triggering command was issued.

Soundness Arguments

- Deductive controller
  - founded on proven model-based reasoning techniques
  - timed language extensions have properties similar to formal real-time specification languages, to allow for straightforward verification
  - algorithms implement a tractable approximation of factored POSMDP semantics
  - despite worst-case exponential performance of on-line reasoning, practical experience has shown adequate performance for typical engineered systems
  - deductive controller enables in-the-loop robustness

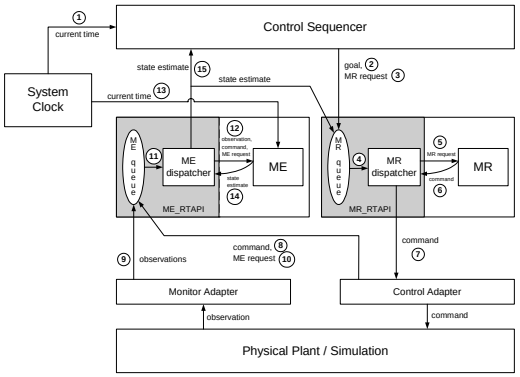
Soundness Arguments (cont.)

- Control Sequencer
  - graphical language for control programs unifies:
    - representational efficiency of Timed Statecharts,
    - executable computational model for, and
    - verifiability properties of formal RT specification languages
  - execution algorithm provides the capabilities of robotic execution languages:
    - conditional execution
    - goal-driven execution
    - closed-loop execution
    - reactive preemption
  - execution algorithm is linear in # of THCA locations
  - implemented algorithm proven to conform to specified control sequencer semantic model

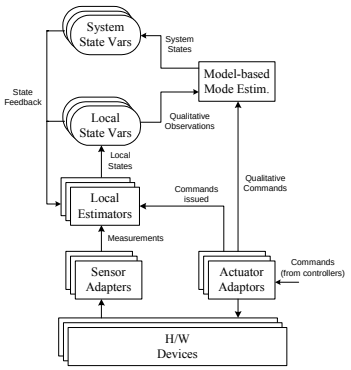
Soundness Arguments (cont.)

- Overall Executive
  - "traditional" model-based control architecture, familiar to spacecraft control and system engineers
  - control program provides "set points" for deductive controller
  - executive reacts to feedback from plant under control
  - modular and expandable architecture
  - can interface with existing system-level planning technologies (e.g. Kirk, ASPEN, EUROPA)

Execution Architecture



ME in MDS



"State of the Art" Solutions

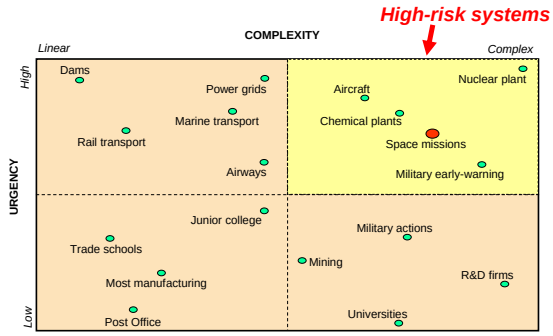
	SCL	ESL	TDL	CIRCA-II	Livingstone/L2	Titan	Timed M-B Exec
Complexity of Plant Interactions					✓	✓	✓
Complexity of Control Spec.	✓	✓	✓	✓		✓	✓
Complexity of Fault Behavior	✓	ad-hoc	ad-hoc	✓	✓	✓	✓
Timed Plant Behavior				✓			✓
Timed Control Spec.	✓	✓	✓	✓			✓

"State of the Art" Solutions

	SCL	ESL	TDL	CIRCA-II	Livingstone/L2	Titan	MDS	Timed M-B Exec
Complexity of Plant Interactions					✓	✓	f/w*	✓
Complexity of Control Spec.	✓	✓	✓	✓		✓	✓	✓
Complexity of Fault Behavior	✓	ad-hoc	ad-hoc	✓	✓	✓	✓	✓
Timed Plant Behavior				✓			f/w	✓
Timed Control Spec.	✓	✓	✓	✓			✓	✓
Executable Visual Spec.					✓	✓		✓

* f/w: provides framework for addressing the issue, but no explicit solution

Motivation



Adapted from Charles Perrow, "Normal Accidents: Living with High-Risk Technologies", 1984.

Control Sequencer Semantics

- input:
  - timed control program *TCP*
  - sequence of plant state estimates
  - sequence of cycle start times from system clock
- output:
  - sequence of config goals
- internally:
  - updates clock variables according to
  - advances current TCP location according to

Deductive Controller Semantics

- input:
  - plant model TPM
  - sequence of config goals
  - sequence of observations
  - sequence of observation times from system clock
- output:
  - sequence of state estimates
  - sequence of control actions
- internally:
  - composition of Mode Estimation and Mode Reconfiguration semantic specifications

"Standard" POSMDP vs. "TCCA" Factored POSMDP

- TCCA model is "Factored":
  - state depends on multiple timer values, not just single "time" parameter
- Fundamental difference due to type of problem each is meant to address
  - Standard POSMDP model for systems where state changes are more frequent than "decision epochs" (opportunities to take an action)
  - TCCA model for composite system where decision epochs are more frequent than state changes

"Standard" POSMDP vs. "TCCA" Factored POSMDP