State-Based Hybrid Control and Symbolic Model Checking

Julia M. B. Braman
NASA-Johnson Space Center
September 16, 2009
Motivation

Aerobot, lower atmosphere probe of Titan, a moon of Saturn (Courtesy NASA/JPL-Caltech)

NASA-JSC’s Lunar Electric Rover (LER) and Chariot Chassis

Control Plan
• Reconfigurable
• Fault Tolerant
• Complex

Testing vs. Formal Methods

J. Braman

September 16, 2009
Outline

- Introduce control architectures & symbolic model checking
  - *Goal networks = Hybrid systems*
- Design for Verification software, *SBT Checker*
  - *Titan Aerobot goal network example*
- State-based symbolic model checker, *InVeriant*
  - *Titan Aerobot goal network example*
- Hybrid Systems
  - *Lunar Electric Rover hybrid system example*
Goal Network Control Programs

- Goal networks
  - Based on Mission Data System (MDS), a control architecture developed at the Jet Propulsion Laboratory
  - Collections of goal trees

- Goals
  - Constraint on state variable
  - Controlled (associated with commands) or passive

Each goal is denoted by
1. Index
2. Parent goal’s index
3. Tactic number
Hybrid systems are collections of discrete sets of continuous dynamics.
Symbolic Model Checking

Model Checking (Spin, NuSMV):
1. Searches over entire state space
2. No continuous states

Symbolic Model Checking (PHAVer, HyTech):
1. Overapproximation
2. Significant complexity issues

State Space

Overapproximated Unsafe Set

Initial Condition

Unsafe Set

September 16, 2009
Safety vs. Liveness

**Safety** tests to see if a specified unsafe set is reachable w/o regards to specific paths

**Liveness** tests whether conditions are reachable with regards to how and how often

Can we reach this state during execution?

Are we eventually always in this state?

This method deals with safety analysis only.

S (stability)
Previous Work

- **Conversion for Verification**
  - AgentSpeak -> Promela, JPF2 (*Bordini et al.*, 2004)
  - MPL -> Livingstone (*Simmons et al.*, 2000)

- **Hybrid Systems Verification**
  - HyTech (*Henzinger et al.*, 1997)
  - PHAVer (*Frehse*, 2005)
  - Uppaal (*Larsen et al.*, 1997)

- **Design for Verification/Correct-by-Design**
  - D4V (*Mehlitz & Penix*, 2003)
  - Control from LTL (*Kress-Gazit et al.*, 2007)
Goal Network Execution

Goal Network:

1. SpeedLimit
2. TransmitData

Passive State Variable Models:

System Health (SH)
- GOOD
- POOR

Satellite Connection (SC)
- GOOD
- POOR

Goal Trees

1. SpeedLimit
    1. SH == GOOD
        - High
    2. SH == POOR
        - Low

2. TransmitData
    1. SC == GOOD
        - High Rate
    2. SC == POOR
        - Low Rate

J. Braman

September 16, 2009
Groups and Transitions

Group: Set of goals active between consecutive time points

Completion Goal: Controlled goal with transition constraint (GetToPoint)

Time Constraint: Bounds on the amount of time spent executing a group

**Groups and Transitions Diagram**

- **Group**: $G_1$, $G_2$
- **SpeedLimit**, **GetToPoint**, **MaintPosition**, **TransmitData**
- **Completion Transitions**
- **Failure Transitions**

**Time Constraint**

1. **SH == GOOD**
   - High
2. **SH == POOR**
   - Low

*September 16, 2009*
Executable Sets of Goals

Properties:
1. All goals in same group
2. All root goals in group
3. Relational goal tree restrictions
4. Compatible
5. Consistent
Conversion Bisimulation

Goal Networks
- Executable set
- Failure (upon goal re-elaboration)
- Completion (upon goal achievement)
- Passive goal constraints
- Controlled goal constraints

Hybrid Systems
- Location
- Transitions
- Invariants
- Flow equations
- Resets

1. SH == GOOD
2. SH == POOR
1. SpeedLimit
2. Location
3. Low

SHG

High
inv: SH == GOOD
ψ : ẋ ≤ v_{High}

Low
inv: SH == POOR
ψ : ẋ ≤ v_{Low}
Titan Aerobot Mission Example

The Titan Aerobot must:

- Fly to a specified area
- Localize to a map and update it with more details
- Fly at a safe altitude
- Maintain some amount of power
- Watch for spontaneous observation opportunities

Unsafe set specification:
1. Power < 10%
2. Position uncertainty is high, ground visibility is low and the altitude is less than the maximum terrain clearance altitude.
Titan Aerobot Goal Trees

1. Power Mgmt
   - 1. Power < 10%
     - 9. Float (X)
   - 2. Power ≥ 10%
     - 10. Power ∈ [10,30]
       - 11. Stationkeep (X)
     - 11. Power ≥ 50%
       - 12. Ceiling (Z = 5)

2. Spont Science
   - 1. SC == NONE
     - 12. SC == NONE
   - 2. SC == MOTION
     - 13. Stationkeep (X)
     - 14. Stationkeep (X)
   - 3. SC == PRECIP
     - 15. Stationkeep (X)
   - 4. SC == CF
     - 16. Stationkeep (X)
     - 17. Ceiling (Z = 5)
     - 18. Stationkeep (X)
     - 19. Ceiling (Z = 5)
Titan Aerobot Goal Trees

1. Maintain Alt
   - 16. CH ≠ POOR
     - 17. SI = LOW
       - 21. Ceiling (Z = 5)
   - 18. CH ≠ POOR
     - 19. LH = GOOD
     - 20. GV = HIGH
     - 21. SI = HIGH
   - 22. CH = FAIR
     - 23. LH = POOR
     - 22. MaxTer (Z = 4)
   - 28. CH = GOOD
     - 29. LH = POOR
     - 24. Min ER (Z = 3)

Sun Intensity (SI)
Camera Health (CH)
LRF Health (LH)
Ground Visibility (GV)
Position Uncertainty (PU)
State-Based Transitions

**Definition:** A goal network (hybrid system) has *state-based transitions* if each state in the passive state space satisfies the passive constraints (invariant) of some executable set (location).

**Passive State Space (SH):**

- **High**
  - \( \text{inv: } SH = \text{GOOD} \)
  - \( \psi : \dot{x} \leq v_{\text{High}} \)
  - SHP

- **Low**
  - \( \text{inv: } SH = \text{POOR} \)
  - \( \psi : \dot{x} \leq v_{\text{Low}} \)
  - SHG

**SBT Checker:** Design for verification tool that checks for unconstrained passive states
State-Based Transitions Example

Camera Health (CH)
- GOOD
- FAIR
- POOR

Position Uncertainty (PU)
- HIGH
- LOW

Speed Limit
1. PU == LOW
   - CH ≠ POOR
     - High
2. PU == HIGH
   - CH == FAIR
     - Low
3. PU == HIGH
   - CH == POOR
     - Stopped

SBT Checker Output:
- (CH == GOOD, PU == HIGH)
- (CH == POOR, PU == LOW)
State-Based Transitions Example

Camera Health (CH)
- GOOD
- FAIR
- POOR

Position Uncertainty (PU)
- HIGH
- LOW

Control States:
- Speed Limit
  - PU == LOW
    - CH != POOR
      - High
  - PU == HIGH
    - CH != POOR
      - Low
  - CH == POOR
    - Stopped

Pu vs. Pu Table:
- LOW
  - 3
  - 1
  - 1
- HIGH
  - 3
  - 2
  - 2

SBT Checker Output: False
**Theorem:** If

1. Each goal tree in a goal network has state-based transitions
2. All controlled constraints are consistent

The goal network has state-based transitions.

**SBT Checker:**
- Design for verification software tool
- Leverages modularity of state-based transition requirement
- Checks individual goal trees/hybrid automata for state-based transitions
- Returns missing passive constraints
SBT Checker Usage

**Command Line:**
1. Create XML file, including
   a) Models of passive state variables used
   b) Invariants of all executable sets of the goal tree
2. Load “SBTChecker” package into Mathematica using following command:
   ```
   Needs["SBTChecker`"];
   ```
3. Store path of XML file in ‘pathfile’ variable
4. Run the SBT Checker using the following command:
   ```
   SBTCheck[pathfile,CSType->GN]
   ```

**GUI:**
1. Load “SBTChecker” package into Mathematica using following command:
   ```
   Needs["SBTChecker`"];
   ```
2. Launch SBT Checker using the following command:
   ```
   SBTCheckGUI
   ```
3. Input models of passive state variables and invariants of executable sets in goal tree
4. Press “Run Check” button
Titan Aerobot Mission Verification

SBT Checker found missing tactic!
Consistent Controlled Constraints

**Rule:** Can only merge goals on $V$ if $c_1 = c_2$

**High inv:** $SH = \text{GOOD} \& \ SC = \text{POOR}$

**Low inv:** $SH = \text{POOR} \& \ SC = \text{GOOD}$

Notice that two states are missing:
1. $SHG \& SCP$
2. $SHP \& SCG$

**Point:** Transitions are not state-based!
Method:
1. Find locations, invariants, dynamical constraints, and resets.
2. Compare unsafe set constraints with each location.
3. Find path for rate-driven, continuous dependent state variables constrained.

Theorem:
If the hybrid system has state-based transitions, the reachability of locations depends only on the reachability of the states of the passive state variables constrained.
Reachability of Unsafe Locations

**Discrete Passive State Variables**

- **Camera Health (CH)**
  - GOOD
  - FAIR
  - POOR

- **Wind Vector (WV)**
  - HIGH
  - LOW

**Continuous, Rate-Driven Passive State Variables**

- **Discrete Power States**
  - 50
  - [30, 50)
  - [10, 30)
  - < 10

**Power Model**

- HiUse
- MedUse
- LoUse
- HiChar
- LoChar
- Neutral

Transitions depend on:
- Position
- Altitude
- Wind Vector
- Sun Intensity

**Is Power < 10 reachable?**

Find path from Power = 100 (initial condition) to Power < 10.
InVeriant Usage

1. Create XML file that includes the following information
   a) Passive state variable models
   b) Merging rules and constraint types for controlled state variables
   c) Goals in each goal tree
   d) Unsafe set
2. Load InVeriant package using the following command:
   \[
   \text{>> Needs["InVeriantSMC`"]};
   \]
3. Store path of XML file in ‘pathfile’ variable
4. Model check the goal network using the following command:
   \[
   \text{>> InVeriant[pathfile,CSType->GN]}
   \]
Titan Aerobot Mission Verification

InVeriant found

- Locations that satisfy unsafe power condition
- Path from initial condition (Power = 100) to unsafe condition (Power < 10)
Benefits of SBT Checker/InVeriant

- Titan system stats:
  - About 600 locations
  - Over 300,000 transitions
  - 11 state variables, 9 passive, 2 controlled
    - Discrete passive state space (no power) is nearly 600,000 states
- Conversion/PHAVer method
  - 5 hours to convert (thousands of transitions)
  - PHAVer did not complete
- SBT Checker/InVeriant
  - 15 minutes to convert
  - 2 minutes to verify

- InVeriant is fast and efficient
  - No transitions
  - Passive state space does not contribute to complexity
- SBT Checker is scalable because of modularity
- State-based transitions imposes structure on control system
  - Good design practice
- Can reason about rate constraints
- Can apply to hybrid systems w/o group structure
Hybrid System Theory

- Classes of hybrid systems can be verified with this approach
  - Automata must have either
    - State-based transitions
    - Completion transitions
  - Automata must run concurrently
  - Automata may be timed (completion transitions)
Lunar Electric Rover

Chariot B Movie Clip
Design Flow

- Design system
  - Check state-based transition
  - Check consistency
    - Model Check
      - Redesign
      - Redesign
      - Redesign
LER Example Problem

Drive
RC == DRIVE

Docking
RC == DOCK

EVA
RC == EVA

Safe
RC == SAFE
LER Example Problem - ECLS

- **Maintain Health**
  - BH == GOOD

- **Warn**
  - BH == FAIR

- **Auto Return**
  - BH == POOR & AO == OFF

- **Auto Return Override**
  - BH == POOR & AO == ON
LER Example Problem - Power

Nominal
Power >= 50

Comm Off
15 <= Power
& Power < 30
& CO == OFF

AR/No Comm
Power < 15 &
CO == OFF &
AO == OFF

AR/C OR
Power < 15 &
CO == ON &
AO == OFF

Power Save
30 <= Power
& Power < 50

Comm OR
Power ∈ [15, 30) &
CO == ON

AR OR/NC
Power < 15 &
CO == OFF &
AO == ON

AR OR/C OR
Power < 15 &
CO == ON &
AO == ON
LER Example Problem - EVA

- Both Closed
  (RHC & LHC) ||
  Power < 30 || (RHO & LHO & (Power < 50 || CMP || CHP))

- Right Open
  RH == OPEN &
  LH == CLOSED & Power >= 30

- Left Open
  LH == OPEN &
  RH == CLOSED & Power >= 30

- Both Open
  LH == OPEN & RH == OPEN &
  Power >= 30 & CM == GOOD &
  CH != POOR
LER Example Problem - Drive

Drive
RC == DRIVE

Car Low
DH == FAIR ||
(DH == GOOD &
DM == CARLOW)

Crab Low
CH == GOOD &
DH == GOOD &
DM == CRABLOW

Path Plan
CH == GOOD &
DH == GOOD &
DM == PP

Safe
DH == POOR &
DM == SAFE

Car High
Power >= 50 &
DH == GOOD &
DM == CARHIGH

Crab High
Power >= 50 & CH
== GOOD & DH ==
GOOD & DM ==
CRABHIGH
LER Example Problem - Docking

Docking
RC == DOCK

Approach
CH == GOOD & Power >= 30 & HH == GOOD & LS == UNLATCH

Docked
Power >= 30 & LS == LATCH

Undock
Power >= 15 & HH == GOOD & LS != LATCH

Latch
CH == GOOD & Power >= 30 & HH == GOOD & LS == CONTACT

Safe
Power < 15 || (LS != LATCH & (CH == POOR || HH == POOR))
LER Example - Docking Redesigned

Docking
RC == DOCK

Approach
CH == GOOD &
Power >= 30 &
HH == GOOD &
LS == UNLATCH

Docked
Power >= 30 &
LS == LATCH

Undock
(P>=15 & HHG &
LS != LATCH) ||
(15 <= Power < 30 &
LSL)

Safe
Power < 15 ||
(LS != LATCH &
(CH == POOR ||
HH == POOR))

Latch
CH == GOOD &
Power >= 30 &
HH == GOOD &
LS == CONTACT
InVeriant finds locations that have inconsistent controlled constraints

Two culprits:
1. “warn” state variable - changed so that merge between no alarm and alarm would result in an alarm
2. “os” state variable - changed so that auto-return would win the merge
Verification of LER Example

Unsafe set: No auto-return during an EVA

Safe!
Conclusions

- Formal Methods useful for analyzing control of fault tolerant systems
- Design for verification is essential to reduce state space concerns (SBT Checker)
- InVeriant model checker leverages properties of state-based design to formally verify a class of hybrid systems
Useful Information

- **SBT Checker:**
  http://www.cds.caltech.edu/~braman/software/SBTChecker.zip
- **InVeriant**
  http://www.cds.caltech.edu/~braman/software/InVeriantSMC.zip
- **Documentation**
  http://www.cds.caltech.edu/~braman
- **LER Example**
  http://www.cds.caltech.edu/~braman/software/LERExample.zip