Institute for Software Integrated Systems
Vanderbilt University

HyTech: A Model Checker for Hybrid Systems

Ming Xiong
James Hill

Motivation

- In mission-critical applications, formal guarantees about the absence of logical and timing errors are desirable
- Time Automata – focus on real-time systems
- Hybrid Automaton – focus on more general hybrid systems

Model-Checking Technology

- Used for system verification
- A formal model of a system is checked, fully automatically, for correctness with respect to a requirement expressed in temporal logic
- Symbolic model checking has been widely used to verify complex systems

Overview of HyTech

- Provides a yes or no to correctness requirement
- Provides diagnostic information that aids in design and debugging, e.g. computes necessary constraints that help finding correct design parameters
- Approximate system using linear hybrid automata

Hybrid Dynamic System

- A dynamic system mixing Boolean-valued variables and real-valued variables, an variant of hybrid system
- Described by

$$\mathbb{B}^m \times \mathbb{R}^n$$

- Example: thermostat

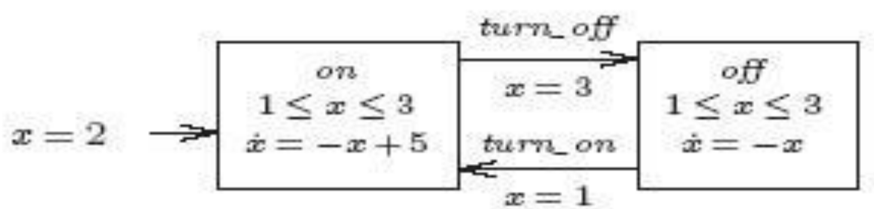

Fig. 1. Thermostat automaton

Hybrid Automata

- A **hybrid automaton** is defined as *H* = (X, V, flow, inv, init, E, jump, e, Σ, syn) where
  - *V* is a set of control modes
  - *X* is a set of continuous variables
  - *Init* is a labeling function that assigns an initial condition to each control mode in *V*
  - *flow* is a labeling function that assigns a flow condition to each control mode in *V*
  - *Inv* is a labeling function that assigns a invariant condition to each control mode in *V*
  - *E* is a collection of control switches
  - *Jump* is a labeling function that assigns a jump condition to each control switch in *E*
  - Σ is a finite set of events
  - *Syn* is a labeling function that assigns an event in Σ to each control switch in *E*

Safety Requirement

- Asserts that nothing bad will happen
- Safety verification amounts to computing the set of reachable states (to see if it's unsafe)
- State assertion
  - a function that assigns to each control in *V* a predicate φ over the variables in *X*
  - the states for which φ is true are called φ-states
    e.g. *inv*-states are precisely admissible states
- A hybrid automata H satisfies the safety requirement specified by *unsafe* if the state assertion *unsafe* is false for all reachable states of H

Linear Hybrid Automata

- Requirements
  - Linearity
  - Flow independence
- Theorem:
  If A is a linear hybrid automaton , and φ is a linear state assertion for A, then Post (φ) can be computed and the result is again a linear state assertion for A
- The above theorem enables safety verification as well as temporal-logic model checking
  - i.e. in HyTech, the model to be checked has to be a linear model

What about non-linear model?

- No direct means of automatically verifying non-linear model
- Has to convert a non-linear model to a linear model
  - Clock translation
  - Linear phase-portrait approximation

Clock Translation

- The idea is sometimes the value of a variable can be determined from a past value (a constant) and the time that has elapsed since the variable had that value
  - Solvability
  - Initialization

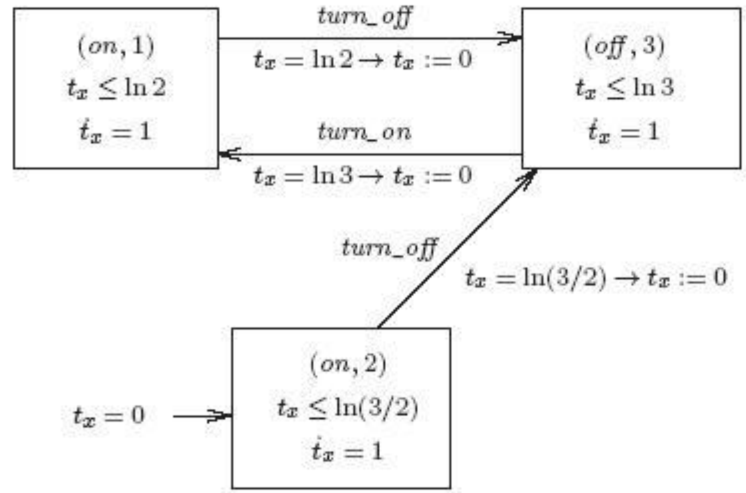

Fig. 3. Clock translation of the thermostat automaton

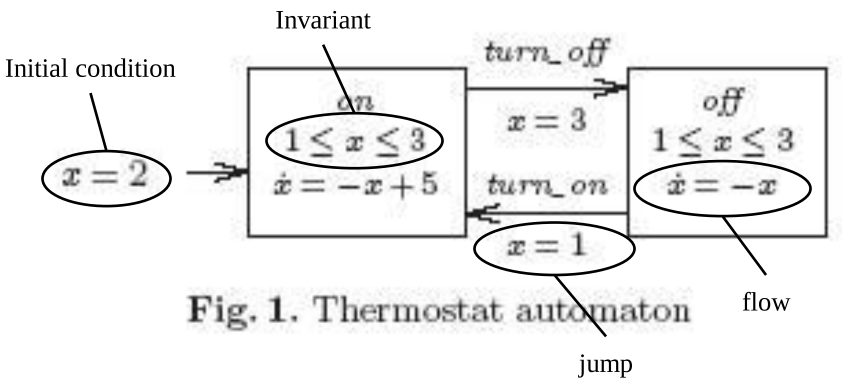

Fig. 1. Thermostat automaton

Linear phase-portrait approximation

- The idea is to relax nonlinear flow, invariant, initial and jump condition using weaker linear condition: each nonlinear predicate p is replaced by a linear predicate

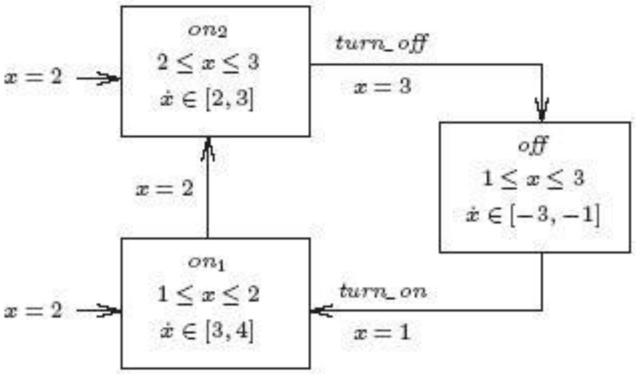

Fig. 5. Tighter linear phase-portrait approximation of the thermostat automaton

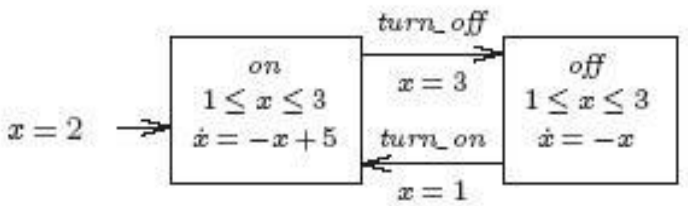

Fig. 1. Thermostat automaton

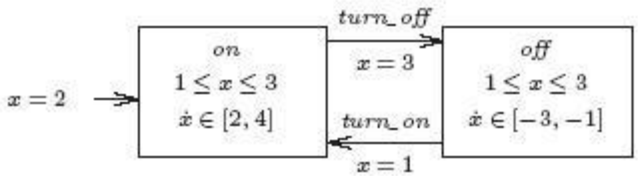

Fig. 4. Linear phase-portrait approximation of the thermostat automaton

Need to be careful about the approximation

Safety Verification for Thermostat systems

- Add extra variables or control modes to specify our safety requirement
- Use both *reach* and *unsafe* assertion
  - if there is any state for which reach and unsafe are true, the safety requirement is violated

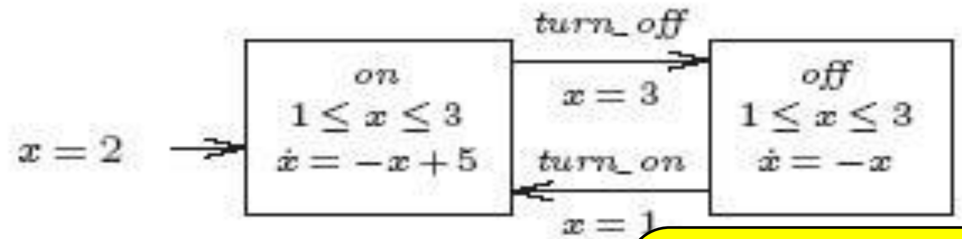

Fig. 1. Thermostat au

Now we can specify y = 60 and z >= 2y/3

Linear phase-portrait approximation

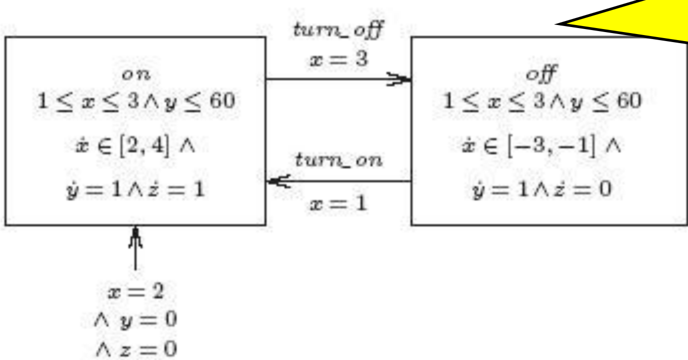

Fig. 6. Linear thermostat automaton for safety verification

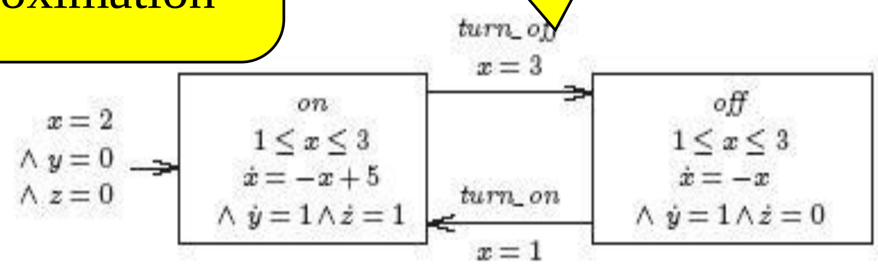

Fig. 2. Thermostat automaton augmented for safety verification

Safety Verification for Thermostat systems (cont'd)

$$\varphi_0 = init = \{(on, x = 2 \wedge y = 0 \wedge z = 0),\ (off, false)\},$$

$$\Downarrow$$

$$\varphi_1 = Post(\varphi_0) = \left\{ \begin{array}{l} (on, x \leq 3 \wedge 2z + 2 \leq x \leq 4z + 2 \wedge y = z), \\ (off, false) \end{array} \right\}.$$

$$\Downarrow$$

$$\varphi_2 = Post(\varphi_1) = \left\{ \begin{array}{l} (on, x \leq 3 \wedge 2z + 2 \leq x \leq 4z + 2 \wedge y = z), \\ (off, x = 3 \wedge \frac{1}{4} \leq z \leq \frac{1}{2} \wedge y = z) \end{array} \right\}.$$

$$\Downarrow$$

$$\varphi_3 = Post(\varphi_2) = \left\{ \begin{array}{l} (on, x \leq 3 \wedge 2z + 2 \leq x \leq 4z + 2 \wedge y = z), \\ (off, 1 \leq x \leq 3 \wedge z + \frac{2}{3} \leq y \leq z + 2 \wedge \\ \quad 2z \leq x \leq 4z) \end{array} \right\}.$$

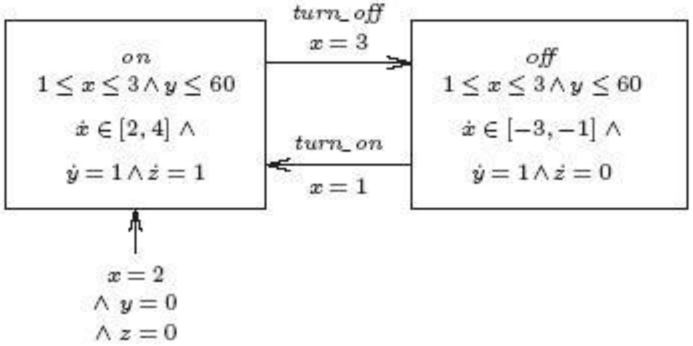

Fig. 6. Linear thermostat automaton for safety verification

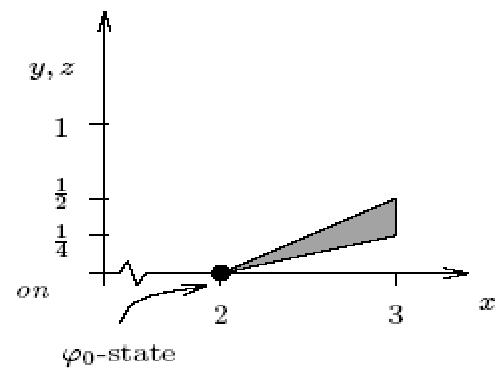

Fig. 8. Flow successors of the φ_0-state

HyTech performs these computations for us, until neither new jump successors nor new flow successors can be found

Parallel Composition

- Sometimes it is convenient to build a separate automaton, called a monitor , whose role is to enter an unsafe state precisely when the original system violates a requirement
- Monitor must observe the original system without changing its behavior

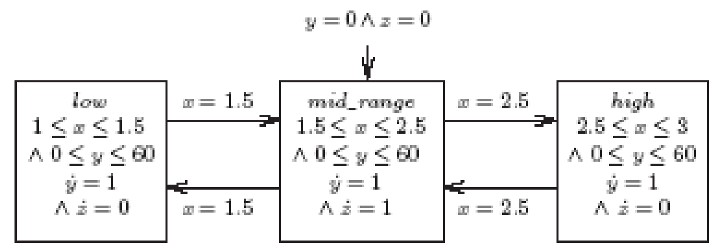

Fig. 9. Monitor automaton

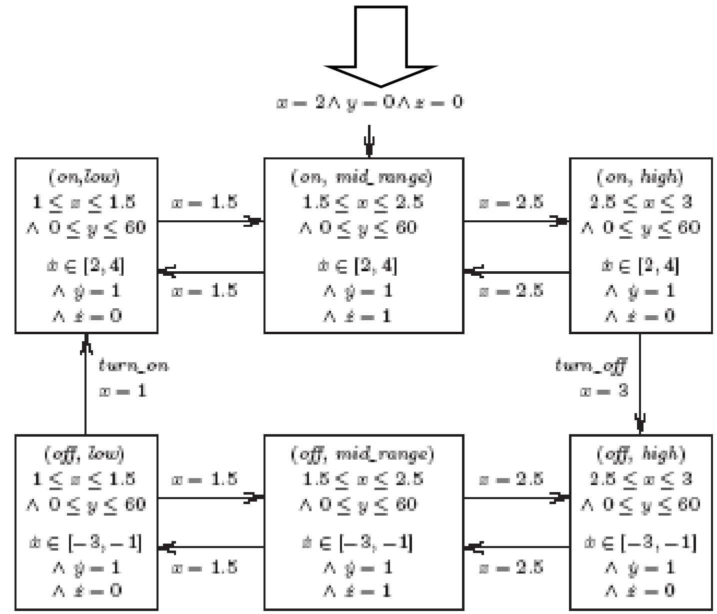

Fig. 10. Parallel composition of thermostat automaton and the