Descriptions of Hybrid Systems

EE219C

Jia Zou

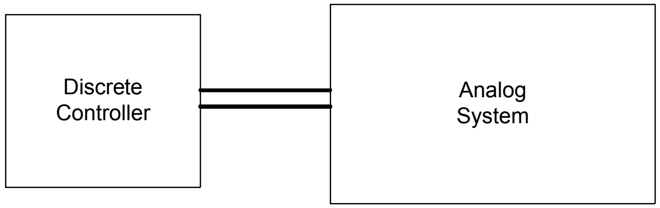
4/25/2007

Outline

- Question:
  - How do we describe hybrid systems?
- One intuitive way to do describe HS
  - Hybrid automata
  - Is this a good idea?
- Other approaches…
  - Lazy linear hybrid automata

What is a Hybrid System

- Discrete program with an analog environment
  - How do we formally verify hybrid systems?
- Modeled as a finite automaton with a set of variables.
  - Vertices => continuous activities
  - Edges => discrete transitions
- H = (Loc, Var, Lab, Edg, Act, Inv)
  - State = (l,v), l є Loc, v є Valuations
  - Stuttering label є Lab
  - (l,a,μ,l’) є Edg
    - An edge is enabled in state (l, v) if for some v’ є V, (v,v’) є μ
    - (l’,v’) is the transition successor of (l,v)



Hybrid System Example

- Leaky gas burner
  - Loc: leak, no leak
  - Var: x, y, z.
  - Inv: x <= 1
  - Transition relation specified by guard
    - μ = {NULL, (x < 30, x >= 30)}

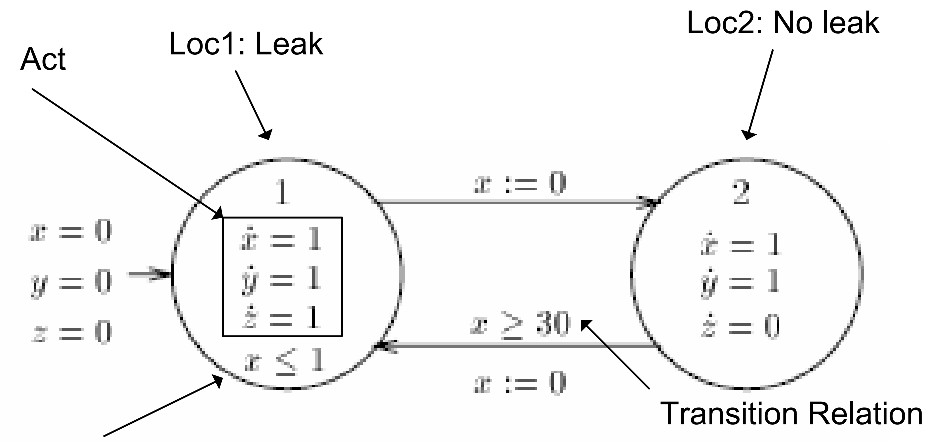


Figure 4: Leaking gas burner

Hybrid System Transitions

- A run [H] of a hybrid system:
  - $\rho : \sigma_0 \mapsto^{t_0}_{f_0} \sigma_1 \mapsto^{t_1}_{f_1} \sigma_2 \mapsto^{t_2}_{f_2} \ldots$
    - $\sigma_i = (l_i, v_i)$
    - $t_i \in \mathbf{R}^{\geq 0}$
    - $f_i \in Act(l_i)$ $\quad f_i \in Inv(l_i)$
    - Properties:
      - If all Act are smooth functions, then all runs are piecewise smooth
      - A run diverges if it's infinite and $\sum_{i \geq 0} t_i \to \infty$

Run of Hybrid System

- Discrete and instantaneous transition of locations.
- Time delay that changes only the value of the variables, according to Act.
- Time-can-progress function to switch between transition-step and time-step

Transition System

- Hybrid system as a transition system:
  - $T_H = (\Sigma, Lab \bigcup \mathbf{R}^{\geq 0}, \rightarrow)$
- Two types of step relations $\rightarrow$
  - Transition-step relation $\rightarrow^a$

$$\frac{(l,a,\mu,l') \in Edg \quad (v,v') \in \mu \quad v \in Inv(l), v' \in Inv(l')}{(l,v) \rightarrow^a (l',v')}$$

  - Time-step relation $\rightarrow^t$

$$\frac{f \in Act(l) \quad f(0) = v \quad \forall 0 \leq t' \leq t. f(t') \in Inv(l)}{(l,v) \rightarrow^t (l,v')}$$

- Time can progress

$$tcp_l[v](t) \Leftrightarrow \forall 0 \leq t' \leq t. \varphi_l[v](t') \in Inv(l)$$

Linear Hybrid Systems

- Act, Inv, Transition relations are linear.
- Special cases:
  - Act(l, x) = 0 for each location. x: discrete variable.
    - All variables discrete ⇔ discrete system
  - μ(e,x) є {0,1} for each transition e є Edg. x: proposition.
    - All variables are propositions ⇔ finite-state system
  - Act(l, x) = 1 for each location l and μ(e,x) є {0,x} for each transition e. x: clock

More About Special Cases

- $Act(l, x) = k$ for each location l and $\mu(e,x) \in \{0,x\}$ for each transition e. x: skewed clock
  - All variables are propositions are skewed clocks ⇔ Multirate timed system.
  - N-rate timed system: skewed clocks proceed at n different rates.
- $Act(l, x) \in \{0,1\}$ for each l && $\mu(e,x) \in \{0,x\}$ for each e. x: integrator.
  - All variables are integrators: integrator system
- $\mu(e,x) = x$ for each e. x: parameter (symbolic constant)

Linear Hybrid System Example

- Leaky gas burner
  - Multirate timed system
    - X: clock that stores time in current location
    - Y: global clock
    - Z: integrator

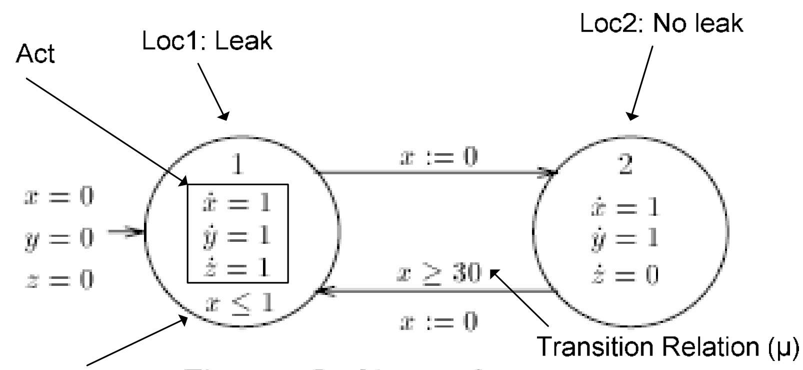


Figure 4: Leaking gas burner

Parallel Composition of HS

- H_1 = (Loc_1, Var, Lab_1, Edg_1, Act_1, Inv_1)
- H_2 = (Loc_2, Var, Lab_2, Edg_2, Act_2, Inv_2)
  - Common set of Var
  - Two hybrid systems synchronized by $Lab_1 \cap Lab_2$
- $H_1 \times H_2$ = ($Loc_1 \times Loc_2$, Var, $Lab_1 \cup Lab_2$, Edg, Act, Inv)
  - $((l_1, l_2), a, \mu, (l'_1, l'_2)) \in Edg$
  - $(l_1, a_1, \mu_1, l'_1) \in Edg_1$ and $(l_2, a_2, \mu_2, l'_2) \in Edg_2$
  - Either $a_1 = a_2 = a$, or $a_1 \notin Lab_2$ and $a_2 = \tau$,
    or $a_2 \notin Lab_1$ and $a_1 = \tau$
  - $\mu = \mu_1 \cap \mu_2$
- $Act(l_1, l_2) = Act_1(l_1) \cap Act_2(l_2)$
- $Inv(l_1, l_2) = Inv_1(l_1) \cap Inv_2(l_2)$
- $[H_1 \times H_2]_{Loc_1} \subseteq [H_1]$ $\qquad$ $[H_1 \times H_2]_{Loc_2} \subseteq [H_2]$

Reachability Problem for Liner Hybrid Systems (LHS)

- A LHS is <u>simple</u> if all local invariants and transition guards are in the form x<=k or k<=x.
- Reachability problem is
  - decidable for simple multirate timed systems.
    - Our previous example
  - Undecidable for 2-rate timed system
  - Undecidable for simple integrator systems

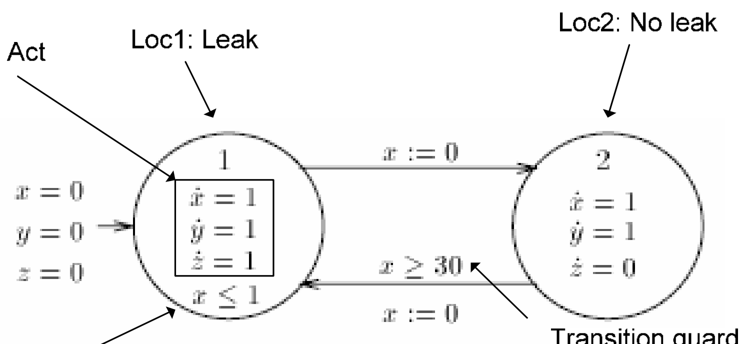
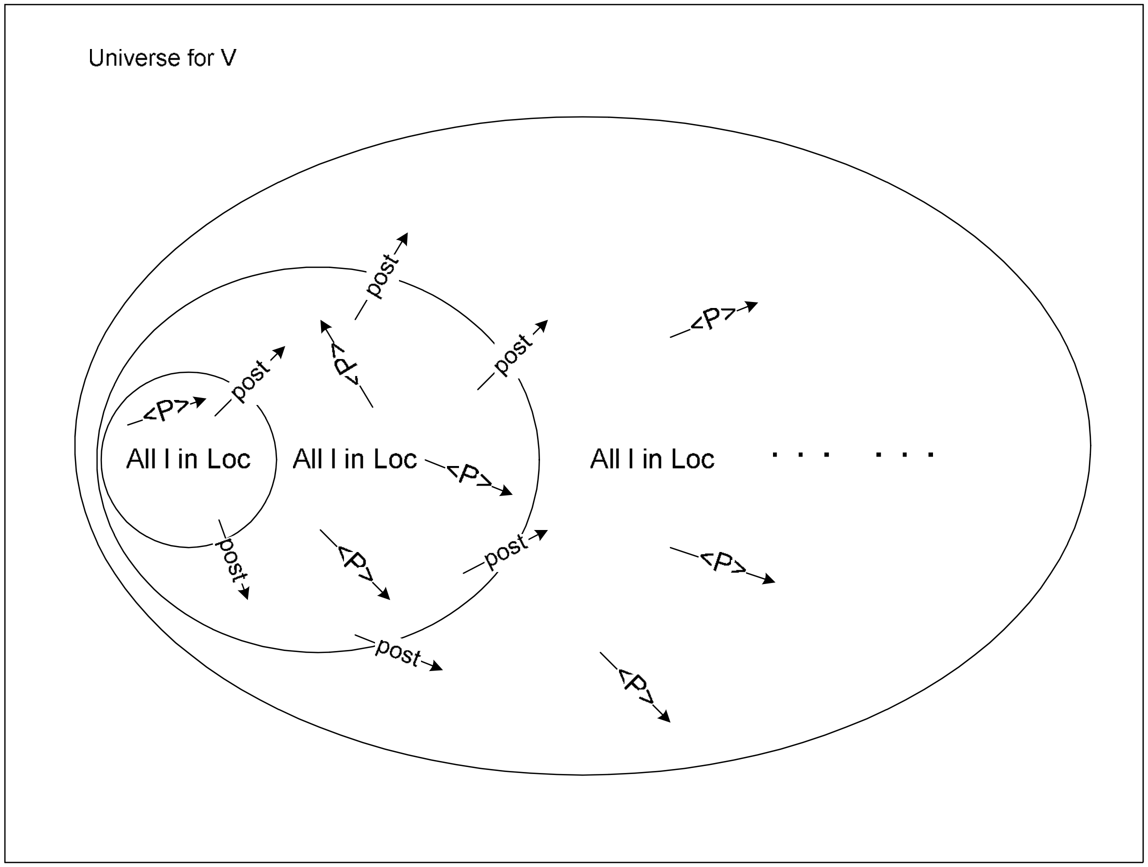


Figure 4: Leaking gas burner

Forward Ananlysis Graphical Representation



Verification of LHS

- Forward Analysis – P is set of valuation
  - Forward time closure of P at l:

$$v' \in \langle P \rangle_l \Leftrightarrow \exists v \in V, t \in \mathbf{R}^{\geq 0}. v \in P \wedge tcp[v](t) \wedge v' = \varphi_l[v](t)$$

  - Postcondition of P with respect to e:

$$v' \in \mathbf{post}_e[P] \Leftrightarrow \exists v \in V, v \in P \wedge (v, v') \in \mu$$

  - A set of states is called a region:

$$\langle R \rangle = \bigcup_{l \in Loc} (l, \langle R_l \rangle_l)$$

$$\mathbf{post}[R] = \bigcup_{e=(l,l') \in Edg} (l', \mathbf{post}_e[R_l])$$

More Forward Analysis

- Symbolic run of linear hybrid system H:

  $\rho = (l_0, P_0)(l_1, P_1)\ldots(l_i, P_i)\ldots \qquad P_{i+1} = \mathbf{post}_{e_i}[\langle P_i \rangle_{l_i}];$

- The region (l_{i+1}, P_{i+1}) is reachable from (l_0, P_0)
- Reachable region $I \mapsto *$

  $\sigma \in (I \mapsto *) \Leftrightarrow \exists \sigma' \in I . \sigma' \mapsto * \sigma.$

- Reachable region of I is the least fixpoint of:

  $X = \langle I \bigcup \mathbf{post}[X] \rangle \qquad X_l = \left\langle I_l \cup \bigcup_{e=(l',l)\in Edg} \mathbf{post}_e[X_{l'}] \right\rangle_l$

- Lemma:
  - If P is a linear set of valuations, then for all l and e, both $\langle P \rangle_l$ and $\mathbf{post}_e[P]$ are linear sets of valuations – makes sure the system is verifiable

Forward Reachability Example

$$\varphi_{1,0} = \langle x = y = z = 0 \rangle_1 = (x \le 1 \wedge y = x = z)$$

$$\varphi_{2,0} = \mathit{false}$$

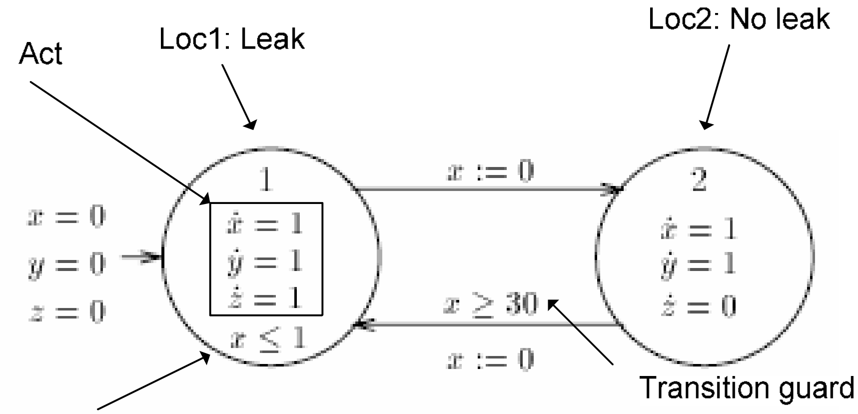


Figure 4: Leaking gas burner

$$\varphi_1 = \langle x = y = z = 0 \vee \mathbf{post}_{(2,1)}[\varphi_2] \rangle_1$$

$$\varphi_2 = \langle \mathit{false} \vee \mathbf{post}_{(1,2)}[\varphi_1] \rangle_2$$

$$\varphi_{1,i} = \varphi_{1,i-1} \vee \langle \mathbf{post}_{(2,1)}[\varphi_2, i-1] \rangle_1$$

$$\varphi_{2,i} = \varphi_{2,i-1} \vee \langle \mathbf{post}_{(1,2)}[\varphi_1, i-1] \rangle_2$$

$$\varphi_{1,1} = \varphi_{1,0} \vee \langle \mathbf{post}_{(2,1)}[\varphi_{2,0}] \rangle_1 = \varphi_{1,0}$$

$$\varphi_{2,1} = \varphi_{2,0} \vee \langle \mathbf{post}_{(1,2)}[\varphi_{1,1}] \rangle_2 = \langle \mathbf{post}_{(1,2)}[x \le 1 \wedge y = x = z = 0] \rangle_2$$

$$= \langle (x = 0 \wedge y \le 1 \wedge z = y \rangle_2 = (z \le 1 \wedge y = z + x)$$

$$\vdots$$

Prove: y>=60 -> 20z <= y

Backward Analysis

- Backward time closure of P at l:

  $$v' \in \langle P \rangle_l \Leftrightarrow \exists v \in V, t \in \mathbf{R}^{\geq 0}.v = \varphi_l[v](t) \wedge v \in P \wedge tcp[v'](t)$$

- Precondition of P with respect to e:

  $$v' \in \mathbf{pre}_e[P] \Leftrightarrow \exists v \in V, v \in P \wedge (v', v) \in \mu$$

- Extension to a region:

  $$\langle R \rangle = \bigcup_{l \in Loc} (l, \langle R_l \rangle_l)$$

  $$\mathbf{pre}[R] = \bigcup_{e=(l',l) \in Edg} (l', \mathbf{pre}_e[R_l])$$

- Initial region I is the least fixpoint of:

  $$X = \left\langle R \bigcup \mathbf{pre}[X] \right\rangle \qquad X_l = \left\langle R_l \cup \bigcup_{e=(l,l') \in Edg} \mathbf{pre}_e[X_{l'}] \right\rangle_l$$

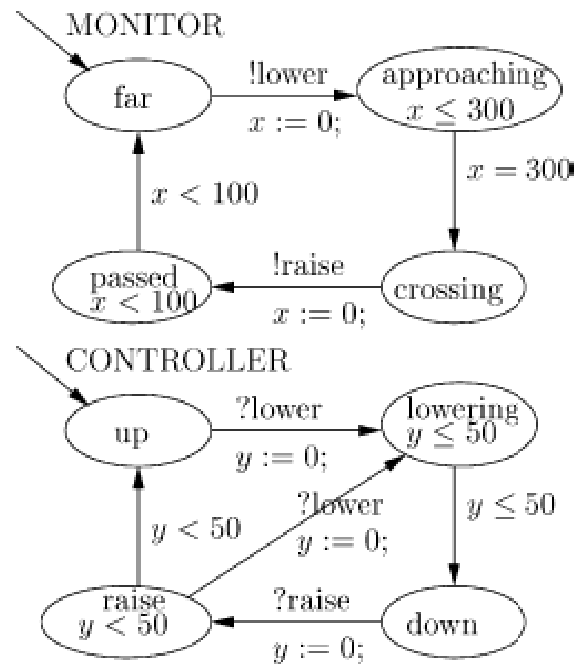
- Lemma:
  - If P is a linear set of valuations, then for all l and e, both $\langle P \rangle_l$ and $\mathbf{pre}_e[P]$ are linear sets of valuations – makes sure the system is verifiable

Description and Specification Languages

- Timed Automata = simple multirate
  - Nondeterministic
  - Does not make transition as long as the Inv are satisfied.
  - PSPACE complexity

Communicating Timed Automata

- Cooperations among processes to construct a state transition
- Channel concept introduced
  - Improve modularity of model description
  - Communicating real-time state machines.
- Monitor + Controller
- No distinction between sender and receiver
  - Model Bus Collisions



The model of gate–monitor–controller.

Hybrid Automata

- Generalization of timed automata
- N-rate timed system
- Undecidable => not subject to algorithmic verification

Logics

- Logic formulas used to describe system behavior
- System description and specifications put into the same language
  - Descriptions as axioms
  - Specification as theorems
- Soundness + completeness check
- Pro:
  - Small models that can prove/disprove theorems quickly
  - Semi-decision procedures that prove first-order logics
- Con:
  - Becomes impossible for large scale systems
  - We can’t build a theorem proving machine in general

Models Dealing With Real-Time Systems

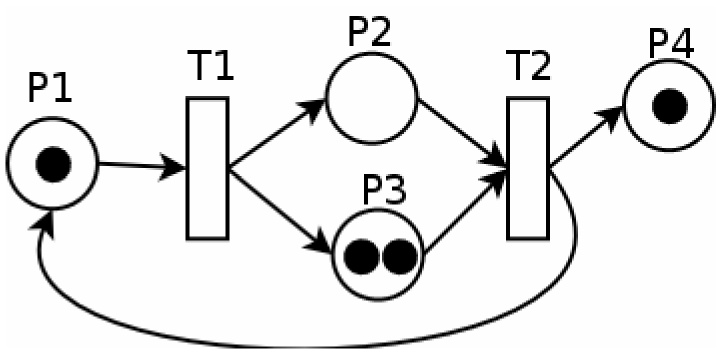
- Case: Train approaching, poles come down
  - Linear-time Propositional Temporal Logic
    - G(approach => F down)
  - LTL with with clock time
    - $\forall x \exists y G((T = x \wedge apprach) \Rightarrow F(T = y \wedge (y - x \leq 300) \wedge down))$
  - Timed Propositional Temporal Logic
    - $Gx.(apprach \Rightarrow Fy.(y - x \leq 300) \wedge down))$
      - Different from LTL with clock
  - Metric Temporal Logic
    - $G(apprach \Rightarrow F_{\leq 300} down)$
  - Asynchronous PTL
    - G[x,y]((x+2)<(y+1))
  - CTL
    - $\forall G(approach \Rightarrow \forall F(down))$
  - TCTL (most used)
    - $\forall G(apprach \Rightarrow \forall F_{\leq 300}(down))$

Timed Process Algebra

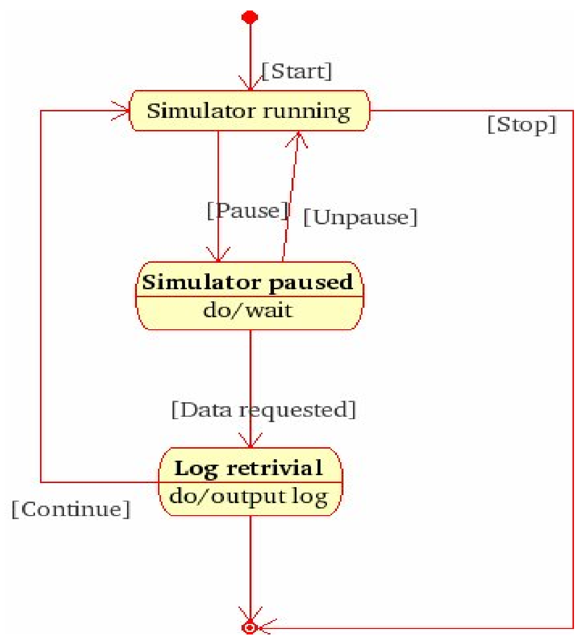
- Three grammar rules
  - Wait t: wait for t time units
  - P_1 t> P_2: P_1, until time t, when no synchronization has happened, then P_2
  - P_1 t↓ P_2: P_1 until time t, no matter what, P_2.

Others

- Timed Petri Nets
  - Places, Tokens, Transitions
  - Many extension to tackle its inexpressiveness

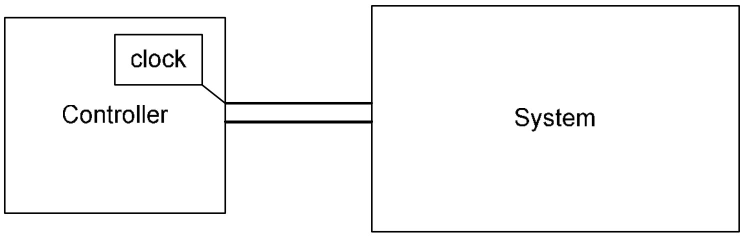


- Statecharts
  - Describe behavioral hierarchies of untimed concurrent systems



Lazy Linear Hybrid Automata

- Definition:
  - A class of LHA where discrete time behavior can be computed and represented as finite state automata.
- Simplifying by sampling.



- Why does this abstraction makes sense?
- Undecidable => decidable?

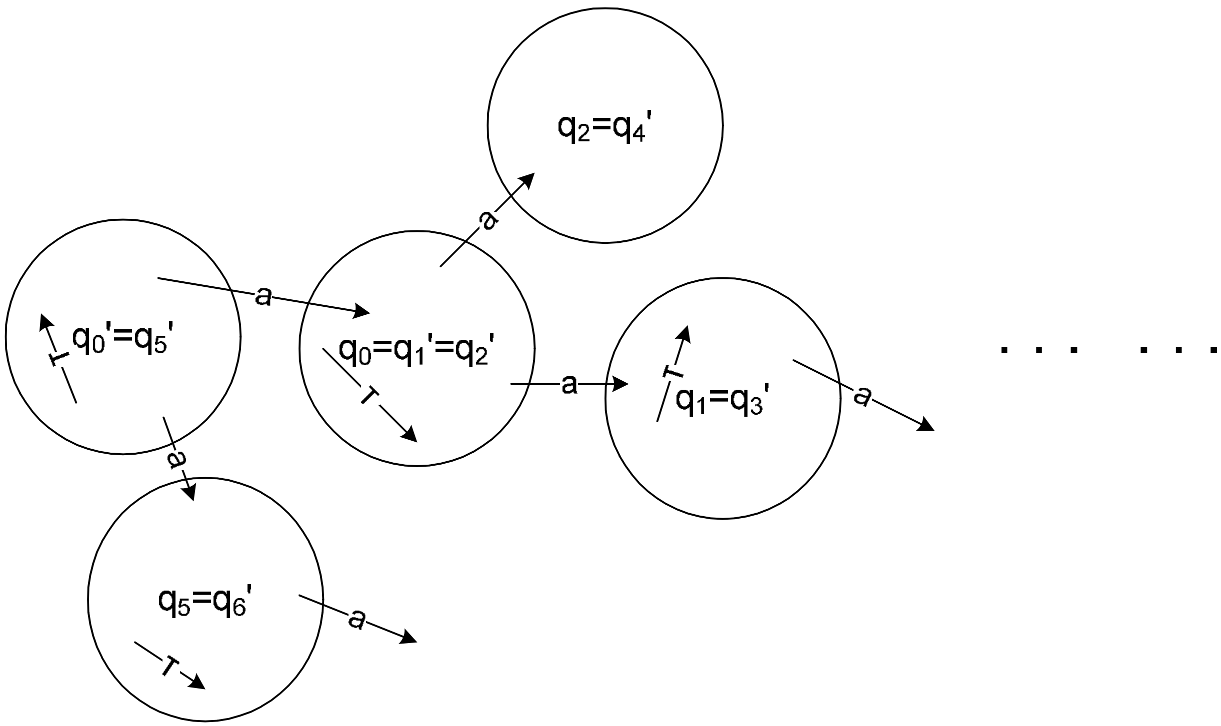
Lazy Linear Hybrid Automata

- Requirements:
  - Periodic sampling
  - Finite precision – bound on the value
- Formulation:
  - On the control side:
    - $A = (Q, Act, q_{in}, V_{in}, D, є, \{p_q\}_{qєQ}, B, =>)$
      - => : (Q x Act X Grd X Q)
      - …D closely related to є?
  - On the system side:
    - Value
    - Guard
    - No states?

Transition Relations

- Configurations:
  - (q,V,q'), q, q' are current and previous control states, V is set of actual values for Var
    - Init: (q_{in}, V_{in}, q_{in}')
    - a : action
    - τ : silent action
    - (q,V,q') =(a)> (q1, V1, q1') iff q1' = q, q=(a, g)>q1
      - t1, t2 are delays. 2 delays to separate two rates
      - Let $v_i = V(i) + \rho_{q'}(i) * t1(i) + \rho_q(i) * (t2(i) - t1(i))$ for each i
        - v_i's satisfies the guards (different from V)
      - $V1(i) = V(i) + \rho_{q'}(i) * t1(i) + \rho_q(i) * (1 - t1(i))$ for each i
    - (q,V,q') =(τ)> (q1, V1, q1') iff q1 = q1' = q, only t1 delay

Transition Relation Graphic representation



More Transition Relations

- With transition relation:
  - Runs can be constructed.
    - $\sigma = (q_0, V_0, q'_0)\alpha_0(q_1, V_1, q'_1)\alpha_1 \ldots (q_k, V_k, q'_k)$
    - Initial condition: (q_0, V_0, q'_0)
    - $\sigma = (q_m, V_m, q'_m) \xrightarrow{\alpha_m} (q_{m+1}, V_{m+1}, q'_{m+1}), \qquad 0 \le m < k$
  - State and act sequences:
    - $st(\sigma) = q_0 q_1 \ldots q_m \ldots q_k \qquad act(\sigma) = \alpha_0 \alpha_1 \ldots \alpha_m \ldots \alpha_k$
  - Languges (set of runs):
    - $L_{st}(A) = \{st(\sigma)\} \qquad L_{act}(A) = \{act(\sigma)\}$
- Claim: The languages are REGULAR subsets of all possible state and act sequences.

Generalizations

- Guards
  - Do not have to be rectangular (not simple)
- Rates of Evolution
  - Does not have to be unique in each control location. Instead can be rectangular.

Conclusion

- Many different approaches (models, languages, etc.) available to solve hybrid systems
- However, most hybrid systems are undecidable, except for some special cases.
- Abstractions may be able to reduce this problem