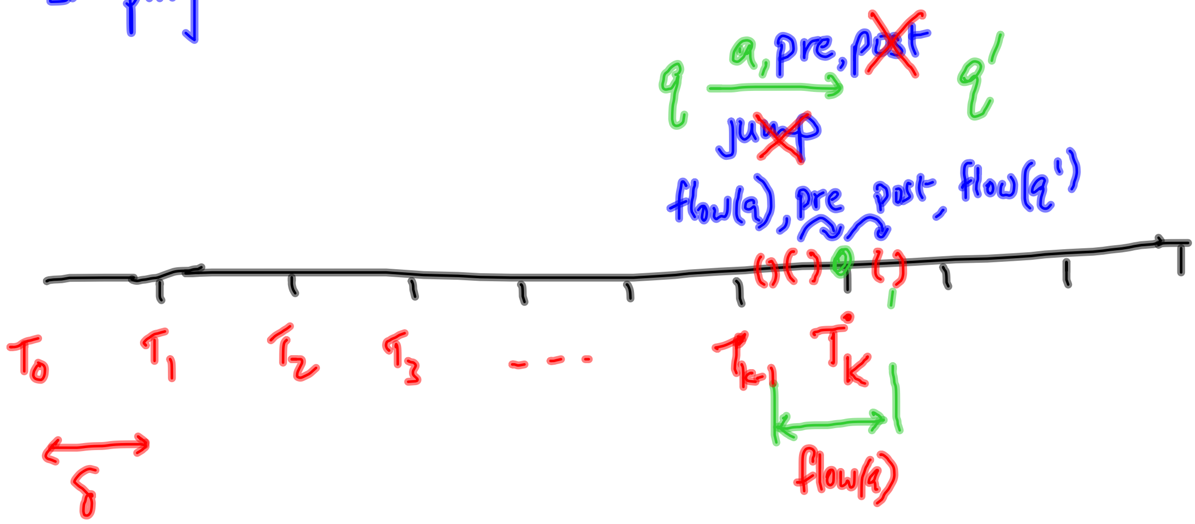
Lazy Hybrid Automata

- Reading values & effecting changes take time
- Sampling values at discrete time intervals.



n vars: $\{x_1, \ldots, x_n\} = X$

$\mathcal{A} = (Q, Act, q_{in}, V_{in}, D, \{\rho_q\}_{q \in Q}, B, \rightarrow)$

delays

flows at q

bounds on value

$\rightarrow \subseteq Q \times Act \times I^n \times Q$

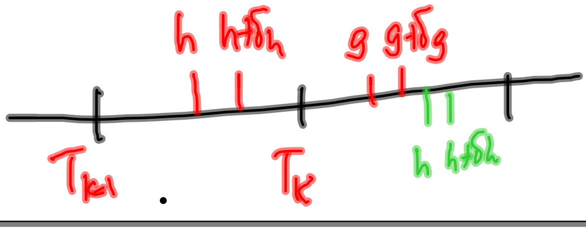
$\delta_q : X \to \mathbb{Q}$

product of n intervals within B

$B = [B_{min}, B_{max}]$

$D = \{g, \delta_g, h, \delta_h\}$

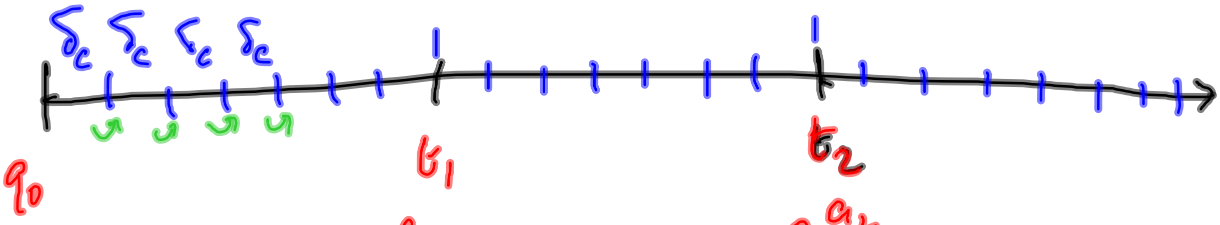
$g < g+\delta_g < h < h+\delta_h$



Simplifying assumptions: $q \xrightarrow{a, I} q'$, $q \neq q'$

Add a silent move $q \xrightarrow{\tau} q'$ where $q = q'$ to denote time elapsing

Delay of one unit δ_c



$q_0 \xrightarrow{a_1} q_1$ $\quad$ $q_1 \xrightarrow{a_2} q_2$

$q_0 \xrightarrow{a_1, t_1} q_1 \xrightarrow{a_2, t_2} q_2 \longrightarrow \cdots$

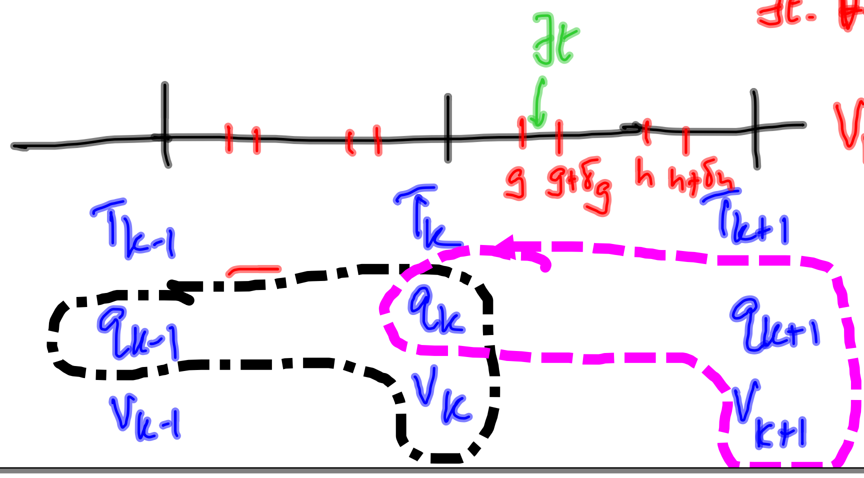
Record values <u>precisely</u> at T_k points

At T_k state : q_k

valuation: $V_k \in B^n$

$\in [g, g+\delta_g]$

$\exists t.\ \forall x_i$



$V_{k+1} = (\delta_{q_{k-1}})(x_i).t$

$+ (\delta_{q_k})(x_i)(1-t)$

$$\begin{array}{ccc} q_k & & q_{k+1} \\ V_k & \xrightarrow{\alpha} & V_k \\ q_{k+1} & \alpha \in Act \cup \{\tau\} & q_k \end{array}$$

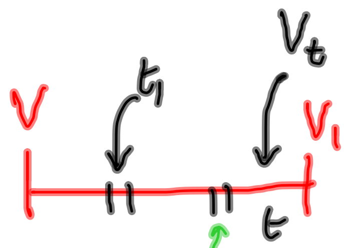
$\alpha = \tau$: V_{k+1} follows from V_k as given before

$q_{k+1} = q_k$

$$(\underbrace{q}_{\text{current}}, V, \underbrace{q'}_{\text{prev}}) \xrightarrow{\alpha \in Act} (\underbrace{q_1}_{\text{current}}, V_1, \underbrace{q_1'}_{\text{prev}})$$

- $q_1' = q$
- $\exists \ q \xrightarrow{\alpha, I} q_1$ s.t.

  I is true at some t_2 in



$$V_{t_2} = V + \underbrace{\delta_{q'}}_{\text{OLD}}(x) \cdot t_1 + \underbrace{\delta_q}_{\text{CURR}}(x)(t_2 - t_1) \text{ satisfies } I$$

$$V_1 = V + \delta_{q'} t_1 + \delta_q (1 - t_1)$$

Start at (q_{in}, V_{in}) $\longrightarrow$ Initial config (q_{in}, V_{in}, q_{in})

Assume $\delta_{q_{in}}$ holds from $t=0$

$$(q_{in}, V_{in}, q_{in}) \xrightarrow{\alpha_1} (q_1, V_1, q_{in}) \xrightarrow{\alpha_2} (q_2, V_2, q_1) \ldots$$

$L_{st}(A)$: state language of A — Regular over Q

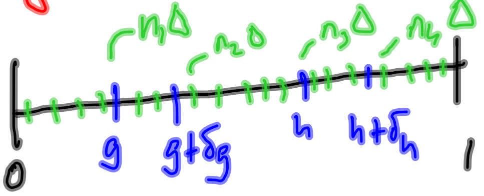
$L_{act}(A)$: action language of A — Regular over $Act \cup \{\tau\}$

Quotient space of configurations (i.e. space of valuations) into a finite number of equivalence classes

Find Γ s.t. all interesting values in evolution of δ are integral multiples of $\Gamma \in \mathbb{Q}$

Start with $D = \{g, h, \delta_g, \delta_h\}$

Δ is largest rational that divides all of these



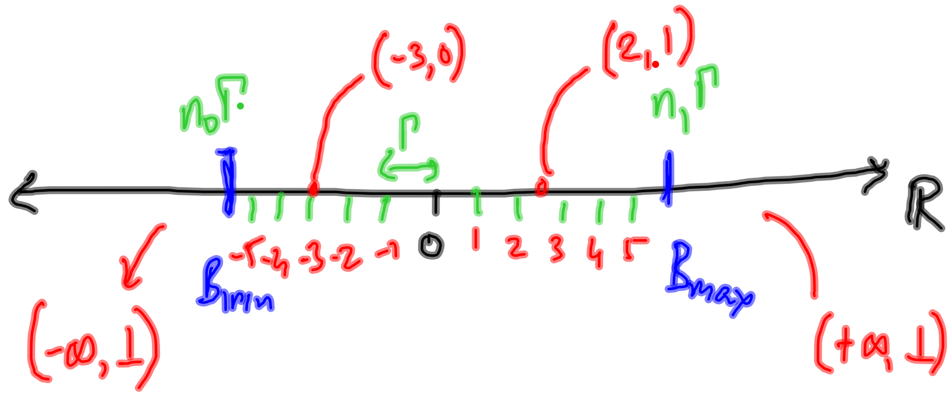
Set Γ to $\gcd \{ \delta_q \cdot \Delta \mid q \in Q, x \in X \}$

Analysis for $X = \{x_1\}$

$$\cup \; \{ B_{min}, B_{max} \}$$

$$\cup \; \{ \ell, r \mid (q, a, [\ell, r], q') \in \rightarrow \}.$$

Given



$V_1 \equiv V_2$ if they agree on this abstraction of the real value of x

$\|V_1\|$ — equivalence class of V_1

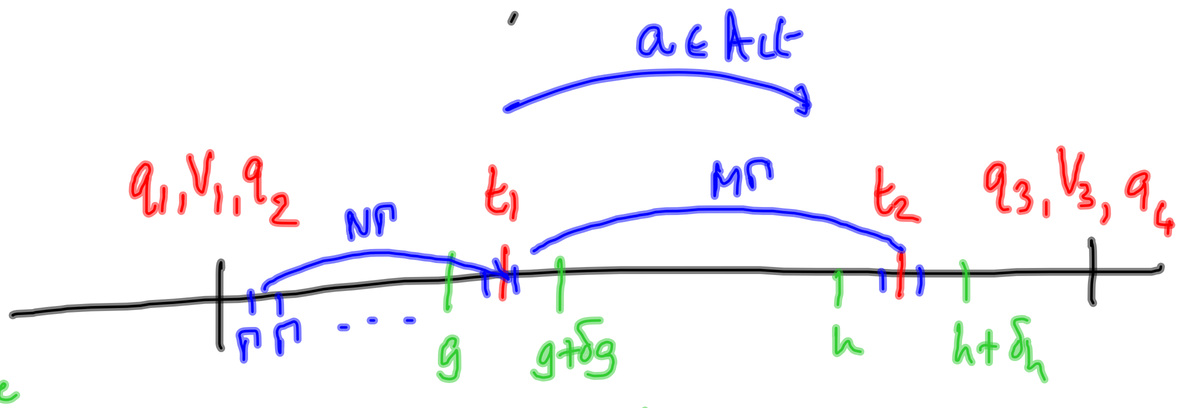
$(q_1, V_1, q_2) \equiv (q_1', V_1', q_2')$ if $q_1 = q_1'$

$q_2 = q_2'$

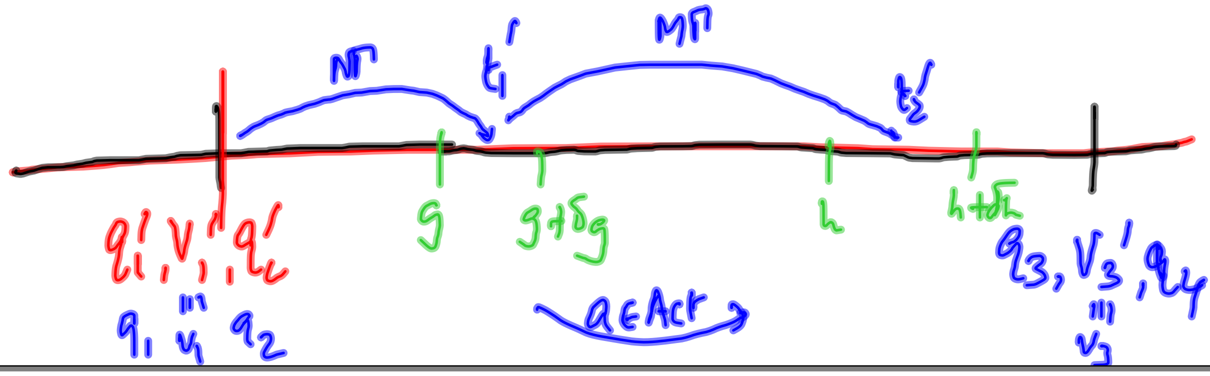
$V_1 \equiv V_1'$

$$
\begin{tikzcd}
c_1 \; (q_1, V_1, q_2) \arrow[d, "\alpha"'] & \equiv & c_1' \; (q_1', V_1', q_2') \arrow[d, "\exists \alpha"'] \\
c_2 = (q_3, V_3, q_4) & & c_2' \; (q_3', V_3', q_4')
\end{tikzcd}
$$

$$c_2 \equiv c_2'$$



Assume determinacy : Fixed transition $q_1 \xrightarrow{a, I} q_3$



Claim: Finite state abstraction captures $L_{st}(\mathcal{A})$, $L_{Act}(\mathcal{A})$

States are $\|(q, V, q')\|$

$$\|(q,V,q')\| \xrightarrow{\alpha} \|(q_1, V_1, q_1')\|$$

Compute this transition relation explicitly for each pair of configurations

Set up linear inequalities to solve for t_1, t_2

For n variables

Set up separate automata for each x_i

Synchronize them

$$(q, \{v_1, v_2, \ldots, v_n\}, q') \xrightarrow{\alpha} (q_1, \{v_1', \ldots, v_n'\}, q_1')$$

if $(q, v_i, q') \xrightarrow{\alpha} (q_1, v_i', q_1')$

for each i

Laziness allows analysis of a larger class

Linear hybrid automata

Back to traces & event structures

Event structure – "tree" of traces

└ explicitly record both $\leq$ & $\#$

causality conflict

Sequential systems

Finite state automata generate "regular trees"

Regular tree:

For each node $s \in T$, let T_s be subtree rooted at s

$s, s' \in T$, $s \approx s'$ if T_s and $T_{s'}$ are isomorphic

T is regular if $\approx$ is of finite index

Regular trees can be "folded" into finite state automata

W. Thomas

Event structures $\approx$ "Trees" of traces

$ES = (E, \leq, \#)$ $(E, \leq)$ is a partial order

$\#$ is inherited via $\leq$

$e_1 \leq e_2,\ e_1 \# e_3 \Rightarrow e_2 \# e_3$

$c \subseteq E$ is a configuration

if $(c \times c) \cap \# = \phi$ &

$c = \downarrow c$

Define ES_c, the event structure "after" c

$$E_c = E - (\{e \mid \exists e' \in c,\ e \# e'\} \cup c)$$

$$\leq_c, \#_c = \leq \cap (E_c \times E_c),\ \# \cap (E_c \times E_c)$$

$c_1 \approx c_2$ if ES_{c_1} is isomorphic to ES_{c_2}

ES is regular if it admits finitely many $\approx$ equivalence classes

Characterize regular event structures?

1-safe Petri net
Asynchronous automata
} → give rise to regular event str.

Converse? Target: 1-safe nets

Thiagarajan's Conjecture:

Thiagarajan's conjecture:

Unlabelled event structure ? $((P,T,F),\lambda)$

$\lambda : T \to \Sigma$

Labelled event structures

$((E,\leq,\#),\varphi)$

Should the resulting net be labelled or unlabelled

$\varphi : E \to \Sigma$, Σ finite

$T = \Sigma$