Stochastic HYPE: modelling stochastic hybrid systems

Vashti Galpin
Laboratory for Foundations of Computer Science
University of Edinburgh

Joint work with Jane Hillston (University of Edinburgh)
and Luca Bortolussi (University of Trieste)

20 April 2011

Introduction

- stochastic hybrid process algebra
  - discrete behaviour
  - continuous behaviour, expressed as ODEs
  - stochastic behaviour, here using exponential distribution

Introduction

- stochastic hybrid process algebra
  - discrete behaviour
  - continuous behaviour, expressed as ODEs
  - stochastic behaviour, here using exponential distribution
- why use a process algebra
  - compositional
  - language allows for abstract reasoning

Introduction

- stochastic hybrid process algebra
  - discrete behaviour
  - continuous behaviour, expressed as ODEs
  - stochastic behaviour, here using exponential distribution
- why use a process algebra
  - compositional
  - language allows for abstract reasoning
- outline
  - HYPE and stochastic HYPE
  - train gate example
  - checking for infinite discrete behaviour
  - I-graphs and acyclicity

Stochastic HYPE model

Stochastic HYPE model

components

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right)$$

Stochastic HYPE model

components

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie}$$

Stochastic HYPE model

$$\overbrace{\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big)}^{\text{components}} \quad \underset{*}{\bowtie} \quad \overbrace{\big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)}^{\text{controllers}}$$

Stochastic HYPE model

components controllers

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right) \quad \underset{*}{\bowtie} \quad \left(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\right)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

Stochastic HYPE model

components — controllers

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right) \quad \underset{*}{\bowtie} \quad \left(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\right)$$

well-defined component

$$C(\mathcal{V}) \stackrel{\mathit{def}}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

components are parameterised by variables

Stochastic HYPE model

components controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j . C(\mathcal{V}) + \underline{\mathsf{init}} : \alpha . C(\mathcal{V})$$

Stochastic HYPE model

components controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

events have event conditions: guards and resets

Stochastic HYPE model

components controllers

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right) \quad \underset{*}{\bowtie} \quad \left(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\right)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

events have event conditions: guards and resets

$\mathrm{ec}(\underline{a}_j) = (f(\mathcal{V}), \mathcal{V}' = f'(\mathcal{V}))$ discrete events

Stochastic HYPE model

components controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

events have event conditions: guards and resets

$\mathrm{ec}(\underline{\mathrm{a}}_j) = (f(\mathcal{V}), \mathcal{V}' = f'(\mathcal{V}))$ discrete events

$\mathrm{ec}(\overline{\mathrm{a}}_j) = (r, \mathcal{V} = f'(\mathcal{V}))$ stochastic events

Stochastic HYPE model

components controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j . C(\mathcal{V}) + \underline{\text{init}} : \alpha . C(\mathcal{V})$$

Stochastic HYPE model

components controllers

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right) \quad \underset{*}{\bowtie} \quad \left(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\right)$$

well-defined component

$$C(\mathcal{V}) \stackrel{\mathit{def}}{=} \sum_j \mathrm{a}_j : \alpha_j . C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha . C(\mathcal{V})$$

influences are defined by a triple

Stochastic HYPE model

components controllers

$$\big(C_1(\mathcal{V}) \bowtie_* \cdots \bowtie_* C_n(\mathcal{V})\big) \quad \bowtie_* \quad \big(Con_1 \bowtie_* \cdots \bowtie_* Con_m\big)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

influences are defined by a triple

$$\alpha_j = (\iota_j, r_j, I(\mathcal{V}))$$

Stochastic HYPE model

components controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)$$

well-defined component

$$C(\mathcal{V}) \stackrel{def}{=} \sum_j \mathrm{a}_j : \alpha_j \,.\, C(\mathcal{V}) + \underline{\mathrm{init}} : \alpha \,.\, C(\mathcal{V})$$

influences are defined by a triple

$$\alpha_j = (\iota_j, r_j, I(\mathcal{V}))$$

influence names are mapped to variables

$$\mathrm{iv}(\iota_j) \in \mathcal{V}$$

Stochastic HYPE model

$$\overbrace{\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right)}^{\text{components}} \quad \underset{*}{\bowtie} \quad \overbrace{\left(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\right)}^{\text{controllers}}$$

Stochastic HYPE model

$$\overbrace{\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big)}^{\text{components}} \quad \underset{*}{\bowtie} \quad \overbrace{\big(\mathit{Con}_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} \mathit{Con}_m\big)}^{\text{controllers}}$$

Stochastic HYPE model

components   controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(\mathit{Con}_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} \mathit{Con}_m\big)$$

controller grammar

Stochastic HYPE model

components controllers

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right) \quad \underset{*}{\bowtie} \quad \left(\mathit{Con}_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} \mathit{Con}_m\right)$$

controller grammar

$$M ::= \mathrm{a}.M \mid 0 \mid M + M$$

Stochastic HYPE model

components

controllers

$$\left(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\right) \quad \underset{*}{\bowtie} \quad \left(\mathit{Con}_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} \mathit{Con}_m\right)$$

controller grammar

$$M ::= \mathrm{a}.M \mid 0 \mid M + M$$

Stochastic HYPE model

components

controllers

$$\big(C_1(\mathcal{V}) \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} C_n(\mathcal{V})\big) \quad \underset{*}{\bowtie} \quad \big(Con_1 \underset{*}{\bowtie} \cdots \underset{*}{\bowtie} Con_m\big)$$

controller grammar

$$M ::= \mathrm{a}.M \mid 0 \mid M + M$$

$$Con ::= M \mid Con \underset{*}{\bowtie} Con$$

Semantics

- HYPE semantics

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{a}$

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{a}$
  - SOS generates labelled transition system (LTS)

    $\langle H, \sigma \rangle \xrightarrow{\underline{a}} \langle H', \sigma' \rangle$

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{a}$
  - SOS generates labelled transition system (LTS)

    $\langle H, \sigma \rangle \xrightarrow{\underline{a}} \langle H', \sigma' \rangle$
  - simple mapping of LTS to hybrid automaton (HA)

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{\mathsf{a}}$
  - SOS generates labelled transition system (LTS)

    $\langle H, \sigma \rangle \xrightarrow{\underline{\mathsf{a}}} \langle H', \sigma' \rangle$
  - simple mapping of LTS to hybrid automaton (HA)
- stochastic HYPE semantics

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{\mathsf{a}}$
  - SOS generates labelled transition system (LTS)

    $\langle H, \sigma \rangle \xrightarrow{\underline{\mathsf{a}}} \langle H', \sigma' \rangle$
  - simple mapping of LTS to hybrid automaton (HA)
- stochastic HYPE semantics
  - mapping to transition driven stochastic HA (TDSHA)

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{\mathsf{a}}$
  - SOS generates labelled transition system (LTS)

    $\langle H, \sigma \rangle \xrightarrow{\underline{\mathsf{a}}} \langle H', \sigma' \rangle$
  - simple mapping of LTS to hybrid automaton (HA)
- stochastic HYPE semantics
  - mapping to transition driven stochastic HA (TDSHA)
  - compositional mapping using TDSHA product on shared events

Semantics

- HYPE semantics
  - no stochastic events, only $\underline{\mathsf{a}}$
  - SOS generates labelled transition system (LTS)

    $\langle H, \sigma \rangle \xrightarrow{\underline{\mathsf{a}}} \langle H', \sigma' \rangle$
  - simple mapping of LTS to hybrid automaton (HA)
- stochastic HYPE semantics
  - mapping to transition driven stochastic HA (TDSHA)
  - compositional mapping using TDSHA product on shared events
  - subset of piecewise deterministic Markov processes (PDMP)

Train gate system

Train gate system

- standard example for hybrid systems modelling

Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing

Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing
- model track and train from 1500m before to 100m after gate

Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing
- model track and train from 1500m before to 100m after gate
- random delay between trains

Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing
- model track and train from 1500m before to 100m after gate
- random delay between trains
- two sensors
  - first: 1000m before crossing sends *approach* signal to controller
  - second: 100m after crossing sends *exit* signal to controller

Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing
- model track and train from 1500m before to 100m after gate
- random delay between trains
- two sensors
  - first: 1000m before crossing sends *approach* signal to controller
  - second: 100m after crossing sends *exit* signal to controller
- gate controller takes time to respond to signals

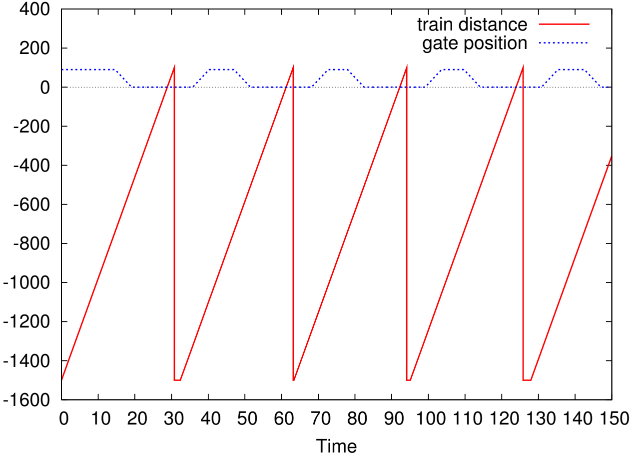
Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing
- model track and train from 1500m before to 100m after gate
- random delay between trains
- two sensors
  - first: 1000m before crossing sends *approach* signal to controller
  - second: 100m after crossing sends *exit* signal to controller
- gate controller takes time to respond to signals
- gate opens and closes at fixed speed

Train gate system

- standard example for hybrid systems modelling
- railway track with guarded crossing
- model track and train from 1500m before to 100m after gate
- random delay between trains
- two sensors
  - first: 1000m before crossing sends *approach* signal to controller
  - second: 100m after crossing sends *exit* signal to controller
- gate controller takes time to respond to signals
- gate opens and closes at fixed speed
- safety property
  - when the train is 100m before the gate, the gate is closed

Train gate system



Reasoning about stochastic hybrid systems

Reasoning about stochastic hybrid systems

- want to ensure certain types of behaviour do not occur

Reasoning about stochastic hybrid systems

- want to ensure certain types of behaviour do not occur
- Zeno behaviour
  - infinite number of events in a finite time period
  - very hard to check

Reasoning about stochastic hybrid systems

- want to ensure certain types of behaviour do not occur
- Zeno behaviour
  - infinite number of events in a finite time period
  - very hard to check
- instantaneous Zeno behaviour
  - infinite number of events in a time instant
  - no continuous behaviour between events
  - not representative of reality
  - not permitted in piecewise deterministic Markov processes

Reasoning about stochastic hybrid systems

- want to ensure certain types of behaviour do not occur
- Zeno behaviour
  - infinite number of events in a finite time period
  - very hard to check
- instantaneous Zeno behaviour
  - infinite number of events in a time instant
  - no continuous behaviour between events
  - not representative of reality
  - not permitted in piecewise deterministic Markov processes
- want to check for this behaviour abstractly
  - without investigating the full behaviour of a model
  - consider controller and event conditions
  - combine these and ignore continuous behaviour

Avoiding instantaneous Zeno behaviour

- basic idea

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events
- I-graph construction

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events
- I-graph construction
  - node: controller state, event, binary activation vector for events, $b_1 \ldots b_n$ for n events

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events
- I-graph construction
  - node: controller state, event, binary activation vector for events, $b_1 \dots b_n$ for n events
  - edge: if transition in LTS with that event *and* event is enabled *and* after vector is an update by event of before vector

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events
- I-graph construction
  - node: controller state, event, binary activation vector for events, $b_1 \ldots b_n$ for n events
  - edge: if transition in LTS with that event *and* event is enabled *and* after vector is an update by event of before vector
  - update of vector by event determines which other events have been activated or disabled by this event

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events
- I-graph construction
  - node: controller state, event, binary activation vector for events, $b_1 \ldots b_n$ for n events
  - edge: if transition in LTS with that event *and* event is enabled *and* after vector is an update by event of before vector
  - update of vector by event determines which other events have been activated or disabled by this event
  - include all controller states, all events, activation vector of 1's

Avoiding instantaneous Zeno behaviour

- basic idea
  - how does the guard and reset of one event affect whether another event can occur immediately that event
  - capture this information in a finite graph using controller info
  - if acyclic then no infinite sequences of events
- I-graph construction
  - node: controller state, event, binary activation vector for events, $b_1 \dots b_n$ for n events
  - edge: if transition in LTS with that event *and* event is enabled *and* after vector is an update by event of before vector
  - update of vector by event determines which other events have been activated or disabled by this event
  - include all controller states, all events, activation vector of 1's
- gives an overapproximation, ignores initial conditions

Controllers for train gate system

- system controller

$$
\begin{aligned}
Con_a &\stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_a \\
Con_l &\stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_l + \underline{\text{lower}}.Con_e \\
Con_e &\stackrel{def}{=} \underline{\text{appr}}.Con_e + \underline{\text{exit}}.Con_r \\
Con_r &\stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_r + \underline{\text{raise}}.Con_a
\end{aligned}
$$

Controllers for train gate system

- system controller

$$
\begin{aligned}
Con_a &\stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_a \\
Con_l &\stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_l + \underline{\text{lower}}.Con_e \\
Con_e &\stackrel{def}{=} \underline{\text{appr}}.Con_e + \underline{\text{exit}}.Con_r \\
Con_r &\stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_r + \underline{\text{raise}}.Con_a
\end{aligned}
$$

- gate internal controller

$$
\begin{aligned}
GC_o &\stackrel{def}{=} \underline{\text{raise}}.GC_o + \underline{\text{lower}}.GC_l \\
GC_l &\stackrel{def}{=} \underline{\text{raise}}.GC_r + \underline{\text{lower}}.GC_l + \underline{\text{closed}}.GC_c \\
GC_c &\stackrel{def}{=} \underline{\text{raise}}.GC_r + \underline{\text{lower}}.GC_c \\
GC_r &\stackrel{def}{=} \underline{\text{raise}}.GC_r + \underline{\text{lower}}.GC_l + \underline{\text{open}}.GC_o
\end{aligned}
$$

Controllers for train gate system

- sequencing of train travel

$$Seq_a \stackrel{def}{=} \underline{\mathrm{appr}}.Seq_p \qquad Seq_p \stackrel{def}{=} \underline{\mathrm{pass}}.Seq_e$$
$$Seq_e \stackrel{def}{=} \underline{\mathrm{exit}}.Seq_f \qquad Seq_f \stackrel{def}{=} \overline{\mathrm{wait}}.Seq_a$$

Controllers for train gate system

- sequencing of train travel

$Seq_a \stackrel{def}{=} \underline{\text{appr}}.Seq_p \qquad Seq_p \stackrel{def}{=} \underline{\text{pass}}.Seq_e$

$Seq_e \stackrel{def}{=} \underline{\text{exit}}.Seq_f \qquad Seq_f \stackrel{def}{=} \overline{\text{wait}}.Seq_a$

- controller states

$Con_x \underset{*}{\bowtie} GC_y \underset{*}{\bowtie} Seq_z$ will be written as $CxGySz$

Controllers for train gate system

- sequencing of train travel

  $Seq_a \stackrel{def}{=} \underline{\text{appr}}.Seq_p \qquad Seq_p \stackrel{def}{=} \underline{\text{pass}}.Seq_e$

  $Seq_e \stackrel{def}{=} \underline{\text{exit}}.Seq_f \qquad Seq_f \stackrel{def}{=} \overline{\text{wait}}.Seq_a$

- controller states

  $Con_x \bowtie_* GC_y \bowtie_* Seq_z$ will be written as $CxGySz$

- continuous part of the system is

  $Gate \bowtie_* Train \bowtie_* Timer_L \bowtie_* Timer_R$

  and remains unchanged by the occurrence of events because of form of well-defined components

Event conditions

- 8 events (other then $\underline{\text{init}}$)

$$
\begin{array}{rcll}
\mathrm{ec}(\underline{\text{appr}}) & = & (D = -1000, & T_L' = 0) \\
\mathrm{ec}(\underline{\text{pass}}) & = & (D = 0, & \mathit{true}) \\
\mathrm{ec}(\underline{\text{exit}}) & = & (D = 100, & T_R' = 0 \wedge D' = -1500) \\
\mathrm{ec}(\overline{\text{wait}}) & = & (\mathit{delay}, & \mathit{true}) \\
\mathrm{ec}(\underline{\text{lower}}) & = & (T_L = 5, & T_L' = 0) \\
\mathrm{ec}(\underline{\text{raise}}) & = & (T_R = 5 & T_R' = 0) \\
\mathrm{ec}(\underline{\text{closed}}) & = & (G = 0, & \mathit{true}) \\
\mathrm{ec}(\underline{\text{open}}) & = & (G = 90, & \mathit{true})
\end{array}
$$

Event conditions

- 8 events (other then $\underline{\text{init}}$)

$$
\begin{aligned}
\mathrm{ec}(\underline{\text{appr}}) &= (D = -1000, & T'_L = 0)\\
\mathrm{ec}(\underline{\text{pass}}) &= (D = 0, & \mathit{true})\\
\mathrm{ec}(\underline{\text{exit}}) &= (D = 100, & T'_R = 0 \wedge D' = -1500)\\
\mathrm{ec}(\overline{\text{wait}}) &= (\mathit{delay}, & \mathit{true})\\
\mathrm{ec}(\underline{\text{lower}}) &= (T_L = 5, & T'_L = 0)\\
\mathrm{ec}(\underline{\text{raise}}) &= (T_R = 5 & T'_R = 0)\\
\mathrm{ec}(\underline{\text{closed}}) &= (G = 0, & \mathit{true})\\
\mathrm{ec}(\underline{\text{open}}) &= (G = 90, & \mathit{true})
\end{aligned}
$$

- let $b_1 \ldots b_8$ be a binary vector where each element is associated with an event using the above ordering

Instantaneous activation graph

- $(CeGlSp, \underline{\text{closed}}, 11110111) \rightarrow (CeGcSp, \underline{\text{pass}}, 11110110)$
  - transition in controller, $CeGlSp \xrightarrow{\text{closed}} CeGcSp$
  - in source node, all events are enabled except lower
  - in target node, lower and open are disabled

Instantaneous activation graph

- $(CeGlSp, \underline{\text{closed}}, 11110111) \rightarrow (CeGcSp, \underline{\text{pass}}, 11110110)$
  - transition in controller, $CeGlSp \xrightarrow{\underline{\text{closed}}} CeGcSp$
  - in source node, all events are enabled except lower
  - in target node, lower and open are disabled
- closed disables open
  - closed occurs when $G = 0$ and G is not reset by closed
  - hence immediately after closed, $G = 0$
  - open occurs when $G = 90$ so disabled immediately after closed

Instantaneous activation graph

- $(CeGlSp, \underline{\text{closed}}, 11110111) \rightarrow (CeGcSp, \underline{\text{pass}}, 11110110)$
  - transition in controller, $CeGlSp \xrightarrow{\underline{\text{closed}}} CeGcSp$
  - in source node, all events are enabled except <u>lower</u>
  - in target node, <u>lower</u> and <u>open</u> are disabled
- <u>closed</u> disables <u>open</u>
  - <u>closed</u> occurs when $G = 0$ and G is not reset by <u>closed</u>
  - hence immediately after <u>closed</u>, $G = 0$
  - <u>open</u> occurs when $G = 90$ so disabled immediately after <u>closed</u>
- <u>closed</u> does not enable <u>lower</u>

Instantaneous activation graph

- $(CeGlSp, \underline{\text{closed}}, 11110111) \rightarrow (CeGcSp, \underline{\text{pass}}, 11110110)$
  - transition in controller, $CeGlSp \xrightarrow{\underline{\text{closed}}} CeGcSp$
  - in source node, all events are enabled except lower
  - in target node, lower and open are disabled
- closed disables open
  - closed occurs when $G = 0$ and G is not reset by closed
  - hence immediately after closed, $G = 0$
  - open occurs when $G = 90$ so disabled immediately after closed
- closed does not enable lower
- lower disables itself: $\mathrm{ec}(\underline{\text{lower}}) = (T_L = 5, T_L' = 0)$

Instantaneous activation graph

- $(CeGlSp, \underline{\text{closed}}, 11110111) \rightarrow (CeGcSp, \underline{\text{pass}}, 11110110)$
  - transition in controller, $CeGlSp \xrightarrow{\underline{\text{closed}}} CeGcSp$
  - in source node, all events are enabled except $\underline{\text{lower}}$
  - in target node, $\underline{\text{lower}}$ and $\underline{\text{open}}$ are disabled
- $\underline{\text{closed}}$ disables $\underline{\text{open}}$
  - $\underline{\text{closed}}$ occurs when $G = 0$ and G is not reset by $\underline{\text{closed}}$
  - hence immediately after $\underline{\text{closed}}$, $G = 0$
  - $\underline{\text{open}}$ occurs when $G = 90$ so disabled immediately after $\underline{\text{closed}}$
- $\underline{\text{closed}}$ does not enable $\underline{\text{lower}}$
- $\underline{\text{lower}}$ disables itself: $\mathrm{ec}(\underline{\text{lower}}) = (T_L = 5, T_L' = 0)$
- $\overline{\text{wait}}$ disables all events because it has duration

Instantaneous activation graph

- $(CeGlSp, \underline{\text{closed}}, 11110111) \rightarrow (CeGcSp, \underline{\text{pass}}, 11110110)$
  - transition in controller, $CeGlSp \xrightarrow{\underline{\text{closed}}} CeGcSp$
  - in source node, all events are enabled except <u>lower</u>
  - in target node, <u>lower</u> and <u>open</u> are disabled
- <u>closed</u> disables <u>open</u>
  - <u>closed</u> occurs when $G = 0$ and G is not reset by <u>closed</u>
  - hence immediately after <u>closed</u>, $G = 0$
  - <u>open</u> occurs when $G = 90$ so disabled immediately after <u>closed</u>
- <u>closed</u> does not enable <u>lower</u>
- <u>lower</u> disables itself: $\mathrm{ec}(\underline{\text{lower}}) = (T_L = 5, T_L' = 0)$
- $\overline{\text{wait}}$ disables all events because it has duration
- straightforward to determine event updates

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events
- theorem: HYPE model with an acyclic I-graph is well-behaved

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events
- theorem: HYPE model with an acyclic I-graph is well-behaved
- results for simple sequential controllers

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events
- theorem: HYPE model with an acyclic I-graph is well-behaved
- results for simple sequential controllers
- two well-behaved controllers with independent events have well-behaved cooperation

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events
- theorem: HYPE model with an acyclic I-graph is well-behaved
- results for simple sequential controllers
- two well-behaved controllers with independent events have well-behaved cooperation
- two well-behaved controllers whose unshared events do not activate events of the other controller have well-behaved cooperation
  - assuming all shared events appear in the cooperation set

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events
- theorem: HYPE model with an acyclic I-graph is well-behaved
- results for simple sequential controllers
- two well-behaved controllers with independent events have well-behaved cooperation
- two well-behaved controllers whose unshared events do not activate events of the other controller have well-behaved cooperation
  - assuming all shared events appear in the cooperation set
- apply results to train gate controller

Well-behaved stochastic HYPE models

- well-behaved: no infinite sequence of simultaneous events
- theorem: HYPE model with an acyclic I-graph is well-behaved
- results for simple sequential controllers
- two well-behaved controllers with independent events have well-behaved cooperation
- two well-behaved controllers whose unshared events do not activate events of the other controller have well-behaved cooperation
  - assuming all shared events appear in the cooperation set
- apply results to train gate controller
- use compositional results rather than working with all controllers

Train gate system: well-behaved?

- Seq_a is well-behaved
  - simple sequential controller
  - one event is stochastic and inhibits all other events
  - also exit inhibits itself and nothing activates it

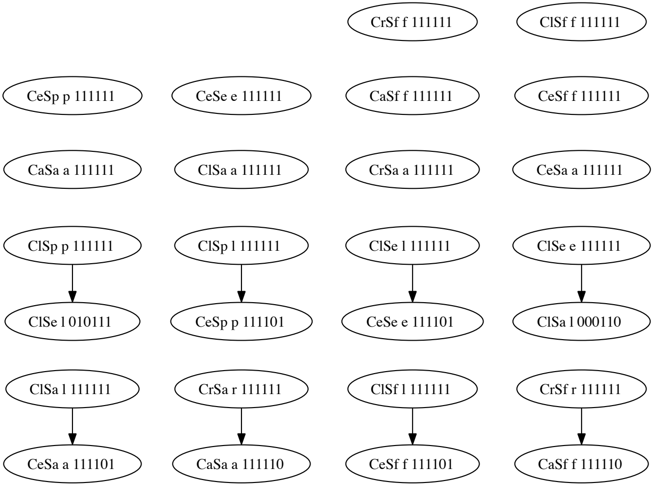
Train gate system: well-behaved?

- Seq_a is well-behaved
  - simple sequential controller
  - one event is stochastic and inhibits all other events
  - also $\underline{\text{exit}}$ inhibits itself and nothing activates it
- Con_a is not well-behaved
  - $Con_l \stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_r$
  - $\underline{\text{appr}}$ does not inhibit itself
  - $(Con_l, \underline{\text{appr}}, 10001) \rightarrow (Con_l, \underline{\text{appr}}, 10001)$

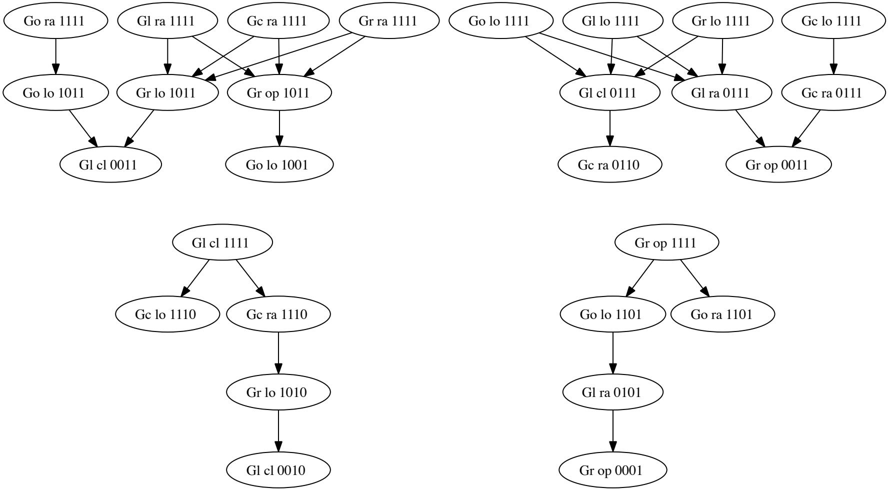
Train gate system: well-behaved?

- Seq_a is well-behaved
  - simple sequential controller
  - one event is stochastic and inhibits all other events
  - also $\underline{\text{exit}}$ inhibits itself and nothing activates it
- Con_a is not well-behaved
  - $Con_l \stackrel{def}{=} \underline{\text{appr}}.Con_l + \underline{\text{exit}}.Con_r$
  - $\underline{\text{appr}}$ does not inhibit itself
  - $(Con_l, \underline{\text{appr}}, 10001) \rightarrow (Con_l, \underline{\text{appr}}, 10001)$
- consider I-graphs of $Con_a \underset{*}{\bowtie} Seq_a$ and $Gate_a$

I-graph of system controller and sequencer



I-graph of gate controller



Train gate system: well-behaved

- $Con_a \bowtie_{*} Seq_a$ is well-behaved

Train gate system: well-behaved

- $Con_a \bowtie_{*} Seq_a$ is well-behaved
- GC_o is well-behaved

Train gate system: well-behaved

- $Con_a \underset{*}{\bowtie} Seq_a$ is well-behaved
- GC_o is well-behaved
- no event activates another event

Train gate system: well-behaved

- $Con_a \underset{*}{\bowtie} Seq_a$ is well-behaved
- GC_o is well-behaved
- no event activates another event
- all shared events are synchronised on

Train gate system: well-behaved

- $Con_a \bowtie_{*} Seq_a$ is well-behaved
- GC_o is well-behaved
- no event activates another event
- all shared events are synchronised on
- hence $Con_a \bowtie_{*} Seq_a \bowtie_{*} GC_o$ is well-behaved

Train gate system: well-behaved

- $Con_a \underset{*}{\bowtie} Seq_a$ is well-behaved
- GC_o is well-behaved
- no event activates another event
- all shared events are synchronised on
- hence $Con_a \underset{*}{\bowtie} Seq_a \underset{*}{\bowtie} GC_o$ is well-behaved
- hence train gate model can be mapped successfully to a piecewise deterministic Markov process

Conclusions

- stochastic HYPE
  - process algebra for stochastic hybrid systems
  - extension of HYPE
  - different underlying semantic model

Conclusions

- stochastic HYPE
  - process algebra for stochastic hybrid systems
  - extension of HYPE
  - different underlying semantic model
- illustrated through a railway gate system

Conclusions

- stochastic HYPE
  - process algebra for stochastic hybrid systems
  - extension of HYPE
  - different underlying semantic model
- illustrated through a railway gate system
- exclusion of infinite behaviour at a time instant
  - instantaneous Zeno behaviour
  - well-behaved stochastic HYPE models
  - construct I-graph from controller and event conditions
  - overapproximation of behaviour
  - check for acyclicity
  - abstract from continuous behaviour

Thank you