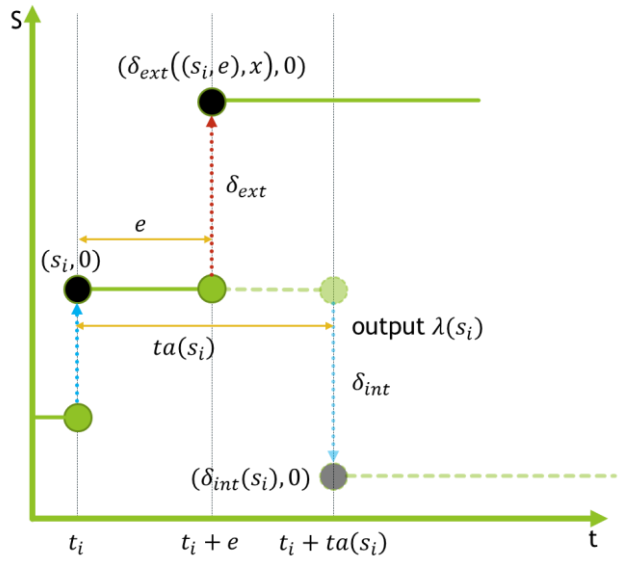


Classic DEVS

Practical Modelling Techniques Using PythonPDEVS

Yentl Van Tendeloo, Hans Vangheluwe

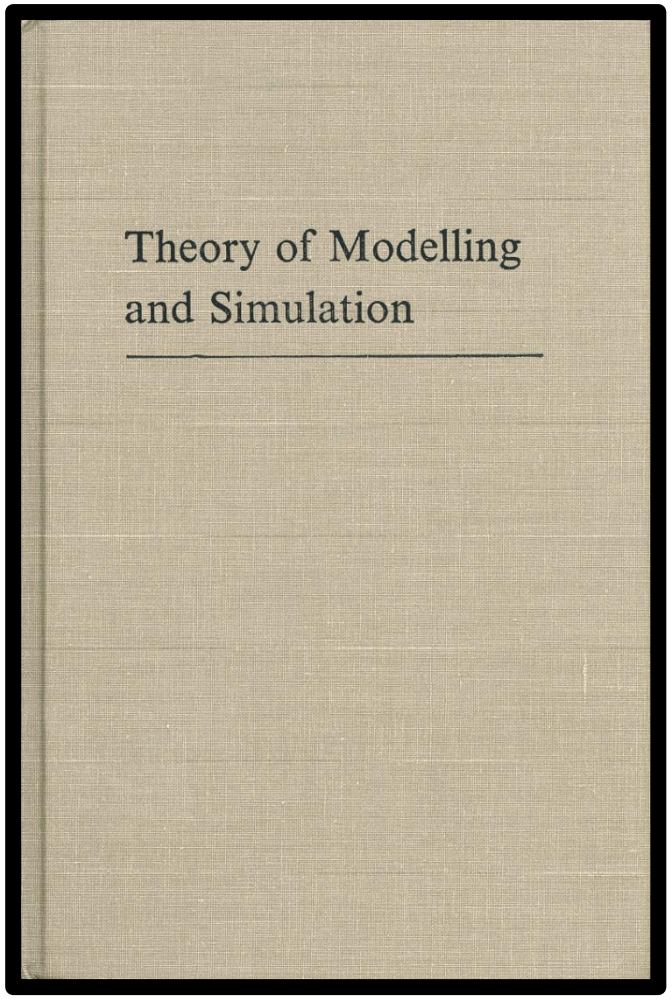
Universiteit Antwerpen

FLANDERS MAKE

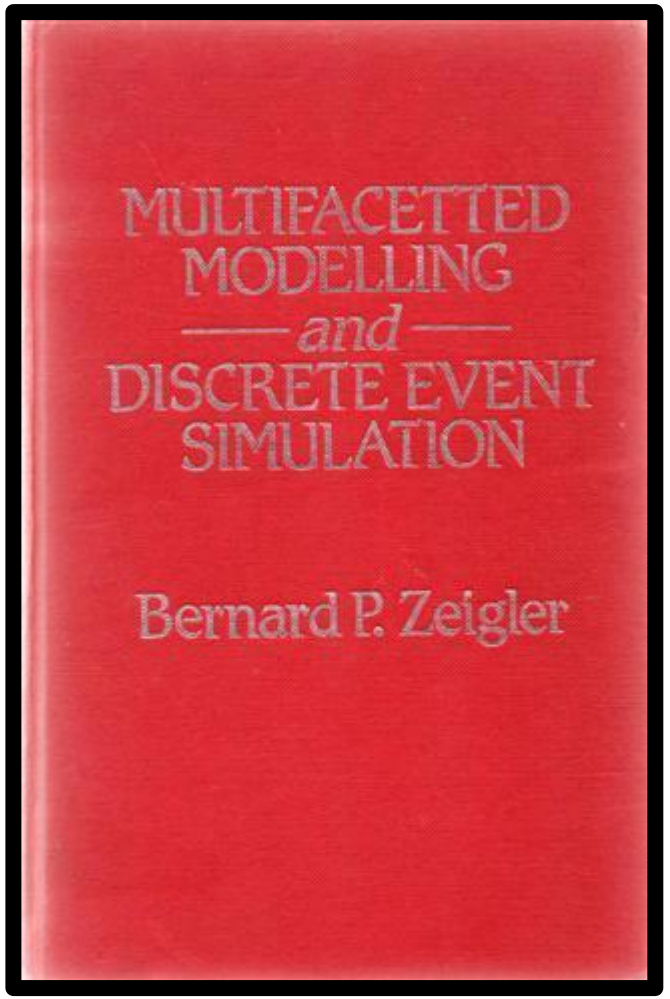
McGill

Introduction

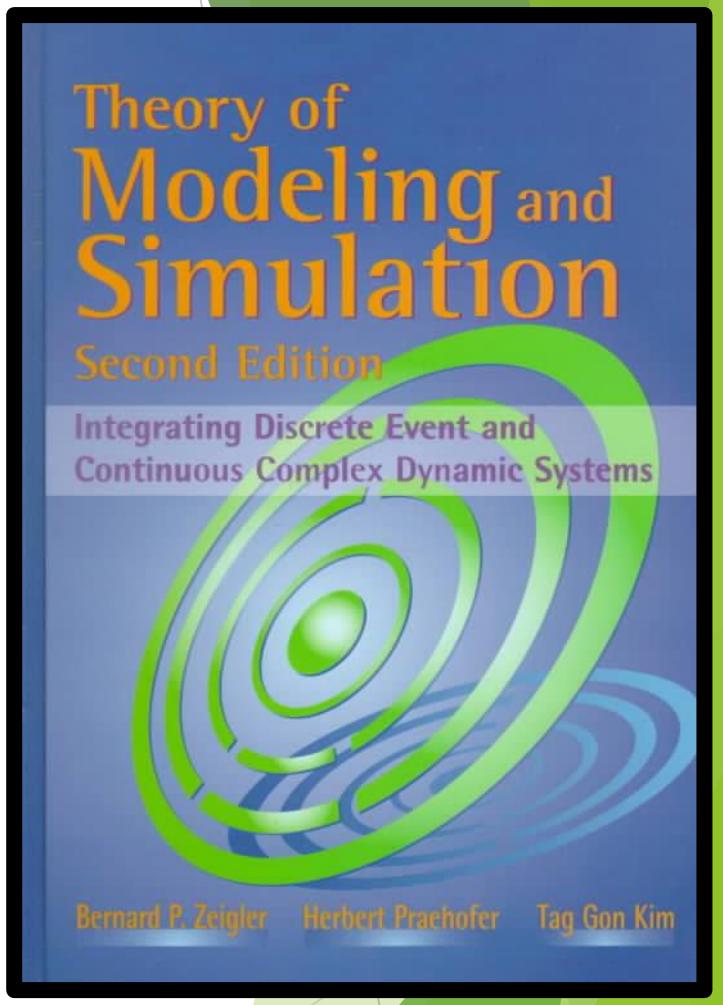
Brief recap of introductory DEVS tutorial



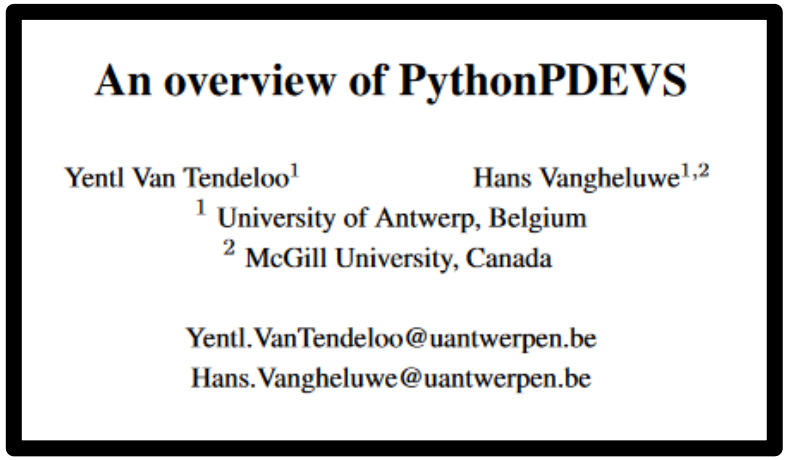
Bernard P. Zeigler.
Theory Of Modelling And Simulation.
1st ed. Wiley, 1976.



Bernard P. Zeigler.
Multifacetted Modelling and Discrete Event Simulation.
1st ed. Academic Press, 1984.



Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim.
Theory Of Modelling And Simulation.
2nd ed. Academic Press, 2000.

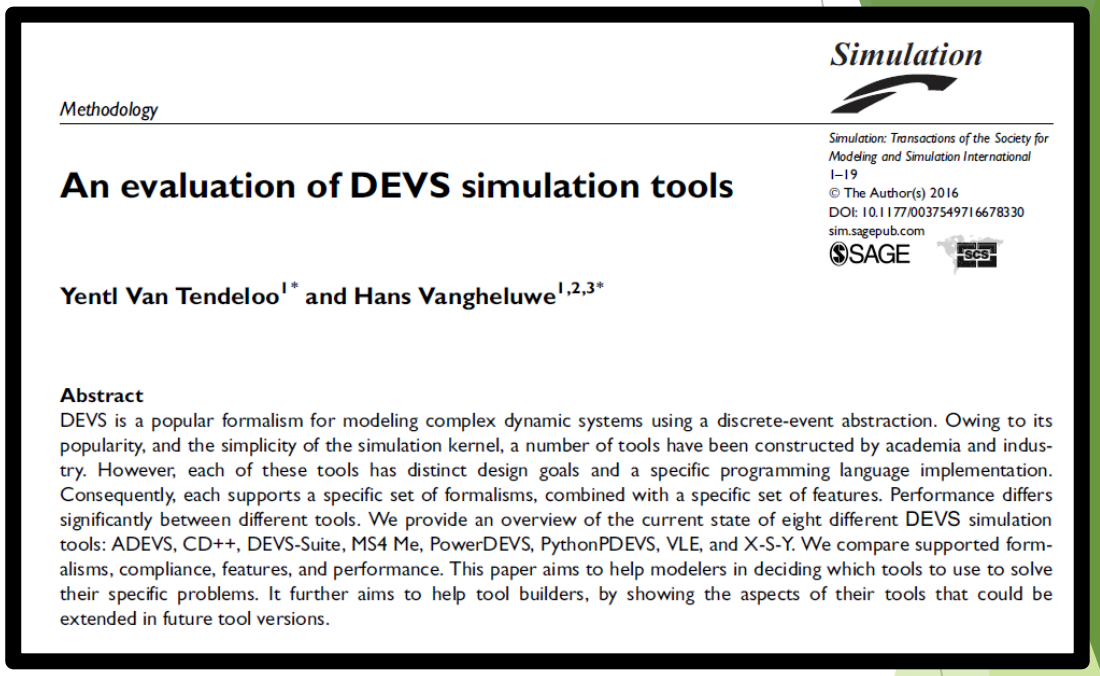


An overview of PythonPDEVS

Yentl Van Tendeloo[1] Hans Vangheluwe[1,2]
[1] University of Antwerp, Belgium
[2] McGill University, Canada

Yentl.VanTendeloo@uantwerpen.be
Hans.Vangheluwe@uantwerpen.be

Yentl Van Tendeloo and Hans Vangheluwe.
An Overview of PythonPDEVS.
In Proceedings of Journées DEVS Francophones (JDF), pages 59-66, 2016.



Simulation

Methodology

Simulation: Transactions of the Society for Modeling and Simulation International
1–19
© The Author(s) 2016
DOI: 10.1177/0037549716678330
sim.sagepub.com
SAGE SCS

An evaluation of DEVS simulation tools

Yentl Van Tendeloo[1]* and Hans Vangheluwe[1,2,3]*

Abstract
DEVS is a popular formalism for modeling complex dynamic systems using a discrete-event abstraction. Owing to its popularity, and the simplicity of the simulation kernel, a number of tools have been constructed by academia and industry. However, each of these tools has distinct design goals and a specific programming language implementation. Consequently, each supports a specific set of formalisms, combined with a specific set of features. Performance differs significantly between different tools. We provide an overview of the current state of eight different DEVS simulation tools: ADEVS, CD++, DEVS-Suite, MS4 Me, PowerDEVS, PythonPDEVS, VLE, and X-S-Y. We compare supported formalisms, compliance, features, and performance. This paper aims to help modelers in deciding which tools to use to solve their specific problems. It further aims to help tool builders, by showing the aspects of their tools that could be extended in future tool versions.

Yentl Van Tendeloo and Hans Vangheluwe.
An Evaluation of DEVS Simulation Tools.
Simulation: Transactions of the Society for Modeling and Simulation International.
2017, 93(2): 103-121

Our presentation uses initialized DEVS models, which contain an initial total state. This was left implicit in the original DEVS specification.

Extending the DEVS Formalism with Initialization Information

Yentl Van Tendeloo* Hans Vangheluwe*‡
{Yentl.VanTendeloo,Hans.Vangheluwe}@uantwerpen.be

DEVS is a popular formalism to model system behaviour using a discrete-event abstraction. The main advantages of DEVS are its rigourous and precise specification, as well as its support for modular, hierarchical construction of models. DEVS frequently serves as a simulation "assembly language" to which models in other formalisms are translated, either giving meaning to new (domain-specific) languages, or reproducing semantics of existing languages. Despite this rigourous definition of its syntax and semantics, initialization of DEVS models is left unspecified in both the Classic and Parallel DEVS formalism definition. In this paper, we extend the DEVS formalism by including an initial total state. Extensions to syntax as well as denotational (closure under coupling) and operational semantics (abstract simulator) are presented. The extension is applicable to both main variants of the DEVS formalism. Our extension is such that it adds to, but does not alter the original specification. All changes are illustrated by means of a traffic light example.

Keywords: Classic DEVS, Parallel DEVS, Experimentation, Initialization

Yentl Van Tendeloo and Hans Vangheluwe. *Extending the DEVS Formalism with Initialization Information*, 2018. ArXiv e-prints.

Atomic DEVS Specification

$$M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

X : *set of input events*

Y : *set of output events*

S : *set of sequential states*

$q_{init} : Q$

$$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$$

$\delta_{int} : S \to S$

$\delta_{ext} : Q \times X \to S$

$\lambda : S \to Y \cup \{\phi\}$

$ta : S \to \mathbb{R}^{+}_{0,+\infty}$

Atomic DEVS Specification

$$M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

X : *set of input events*

Y : *set of output events*

S : *set of sequential states*

$q_{init} : Q$

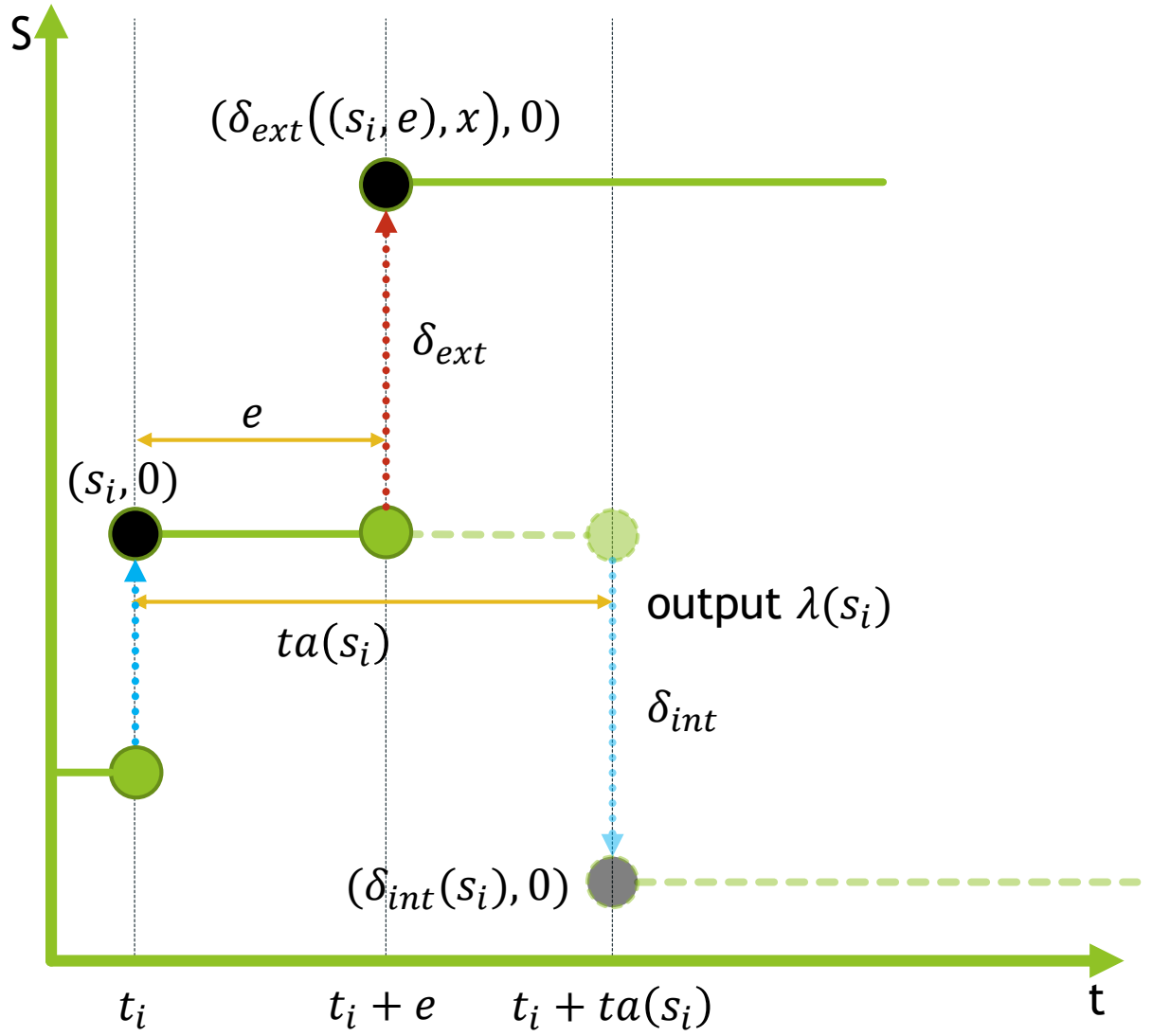
$$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$$

$\delta_{int} : S \to S$

$\delta_{ext} : Q \times X \to S$

$\lambda : S \to Y \cup \{\phi\}$

$ta : S \to \mathbb{R}^{+}_{0,+\infty}$


S
$(\delta_{ext}((s_i, e), x), 0)$
δ_{ext}
e
$(s_i, 0)$
output $\lambda(s_i)$
$ta(s_i)$
δ_{int}
$(\delta_{int}(s_i), 0)$
t
t_i
$t_i + e$
$t_i + ta(s_i)$

Coupled DEVS Specification

$$C = \langle X_{self}, Y_{self}, D, MS, IS, ZS, select \rangle$$

$$MS = \{M_i | i \in D\}$$

$$M_i = \langle X_i, Y_i, S_i, q_{init,i}, \delta_{int,i}, \delta_{ext,i}, \lambda_i, ta_i \rangle, \forall\, i \in D$$

$$IS = \{I_i | i \in D \cup \{self\}\}$$

$$\forall\, i \in D \cup \{self\} : I_i \subseteq D \cup \{self\}$$

$$\forall\, i \in D \cup \{self\} : i \notin I_i$$

$$ZS = \left\{Z_{i,j} \middle| i \in D \cup \{self\}, j \in I_i\right\}$$

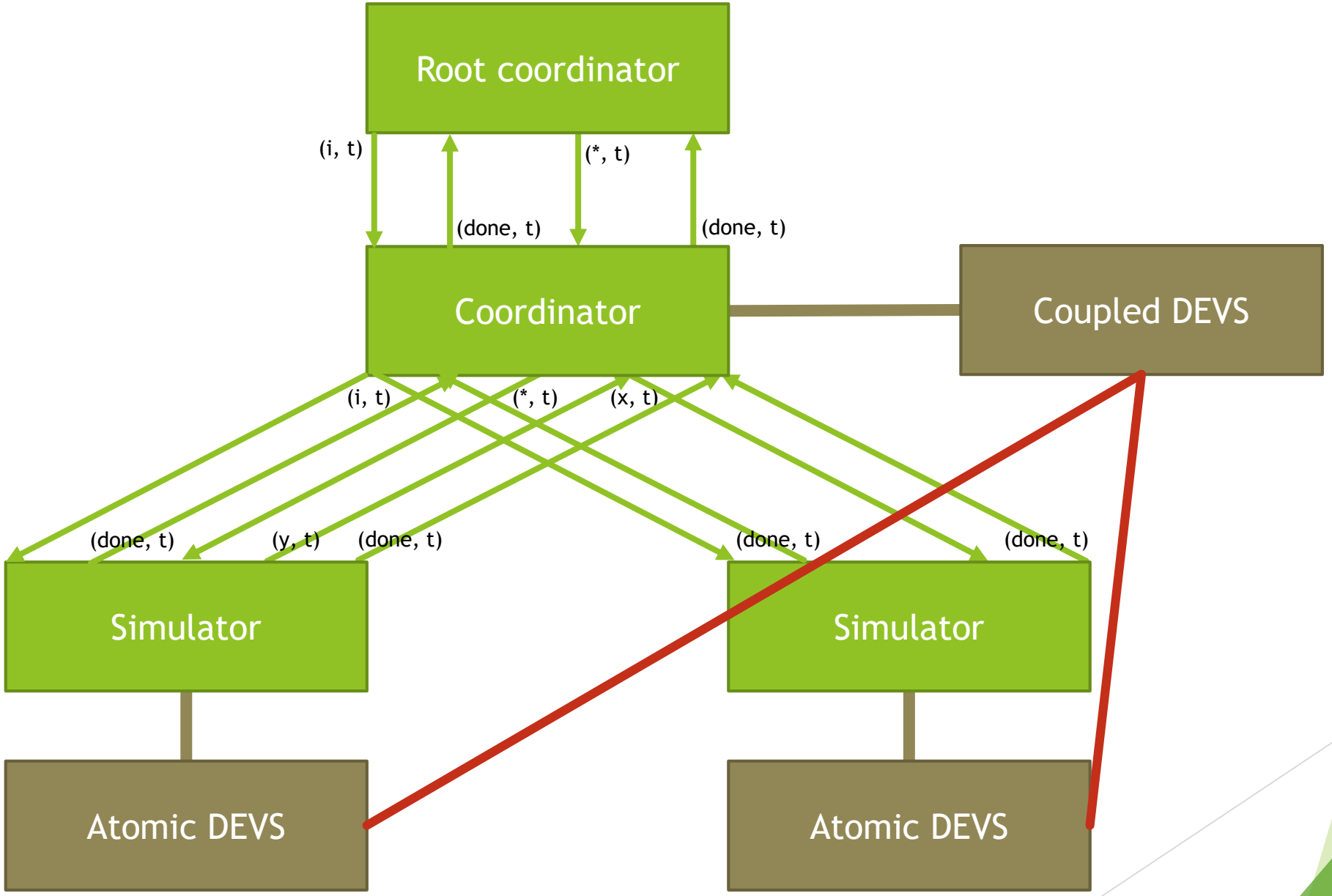
$$Z_{self,j} : X_{self} \to X_j, \forall\, j \in D$$

$$Z_{i,self} : Y_i \to Y_{self}, \forall\, i \in D$$

$$Z_{i,j} : Y_i \to X_j, \forall\, i, j \in D$$

$$select : 2^D \to D$$

$$\forall\, E \subseteq D, E \neq \emptyset : select(E) \in E$$


Root coordinator
(i, t)
(*, t)
(done, t)
(done, t)
Coordinator
Coupled DEVS
(i, t)
(*, t)
(x, t)
(done, t)
(y, t)
(done, t)
(done, t)
(done, t)
Simulator
Simulator
Atomic DEVS
Atomic DEVS

message m	**simulator**	**coordinator**
$(*, from, t)$	simulator correct only if $t = t_N$	
	$y \leftarrow \lambda(s)$	**send** $(*, self, t)$ to i^*, where
	if $y \neq \phi$:	$i^* = select(imm\_children)$
	send $(\lambda(s), self, t)$ to $parent$	$imm\_children = \{i \in D \mid M_i.t_N = t\}$
	$s \leftarrow \delta_{int}(s)$	$active\_children \leftarrow active\_children \cup \{i^*\}$
	$t_L \leftarrow t$	
	$t_N \leftarrow t_L + ta(s)$	
	send $(done, self, t_N)$ to $parent$	

message m	**simulator**	**coordinator**
$(x, from, t)$	simulator correct only if $t_L \leq t \leq t_N$ (ignore δ_{int} to resolve a $t = t_N$ *conflict*)	
	$e \leftarrow t - t_L$	$\forall i \in I_{self}$:
	$s \leftarrow \delta_{ext}(s, e, x)$	**send** $(Z_{self,i}(x), self, t)$ to i
	$t_L \leftarrow t$	$active\_children \leftarrow active\_children \cup \{i$
	$t_N \leftarrow t_L + ta(s)$	
	send $(done, self, t_N)$ to $parent$	

message m	**simulator**	**coordinator**
$(y, from, t)$		$\forall i \in I_{from} \setminus \{self\}$:
		send $(Z_{from,i}(y), from, t)$ to i
		$active\_children \leftarrow active\_children \cup \{i\}$
		if $self \in I_{from}$:
		send $(Z_{from,self}(y), self, t)$ to $parent$
$(done, from, t)$		$active\_children \leftarrow active\_children \setminus \{from\}$
		if $active\_children = \emptyset$:
		$t_L \leftarrow t$
		$t_N \leftarrow min\{M_i.t_N \mid i \in D\}$
		send $(done, self, t_N)$ to $parent$

$t \leftarrow t_N$ of topmost coordinator

repeat until $t \geq t_{end}$ (or some other termination condition)

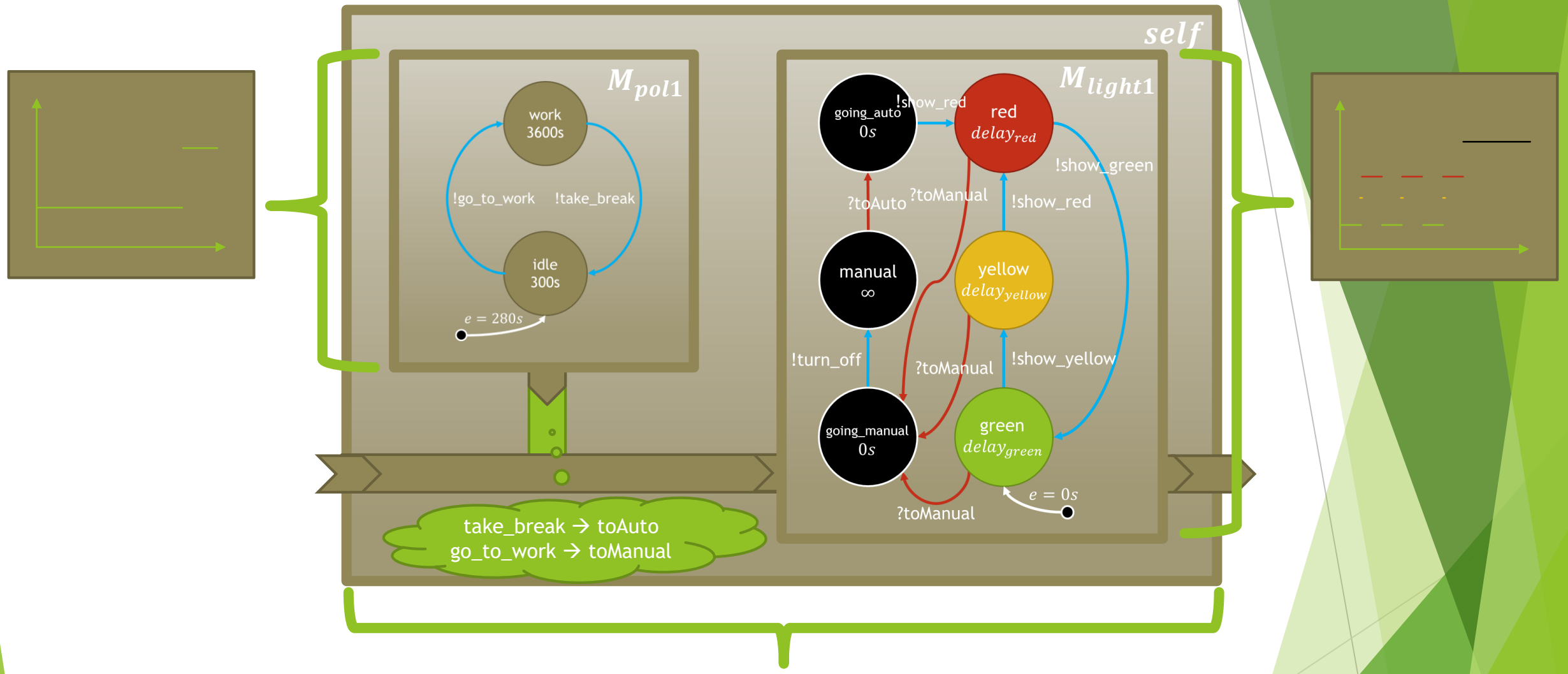
send $(*, main, t)$ to topmost coupled model top

wait for $(done, top, t_N)$

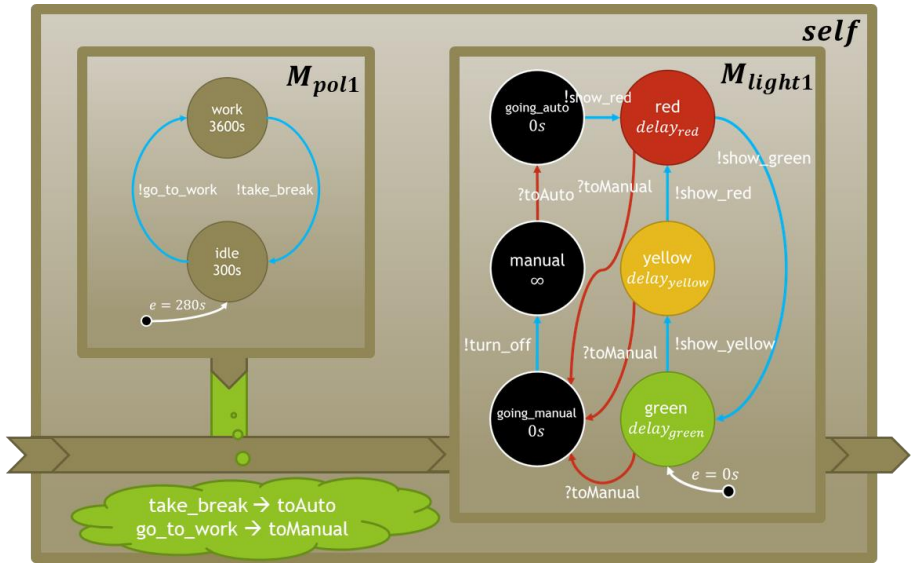
$t \leftarrow t_N$

Closure Under Coupling

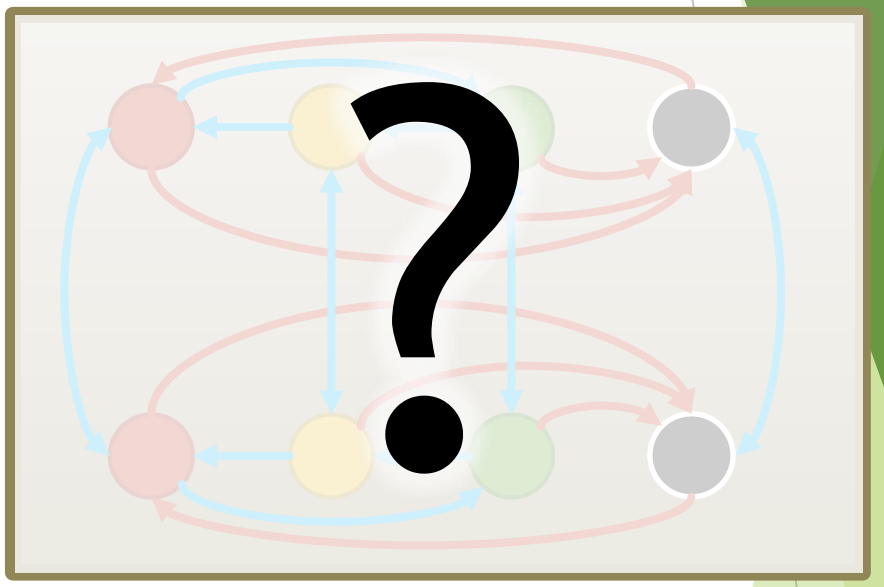
Denotational semantics for Coupled DEVS models


self
M_{pol1}
work
3600s
!go_to_work
!take_break
idle
300s
$e = 280s$
M_{light1}
going_auto
$0s$
!show_red
red
$delay_{red}$
!show_green
?toAuto
?toManual
!show_red
manual
∞
yellow
$delay_{yellow}$
!turn_off
?toManual
!show_yellow
going_manual
$0s$
green
$delay_{green}$
?toManual
$e = 0s$
take_break → toAuto
go_to_work → toManual

?

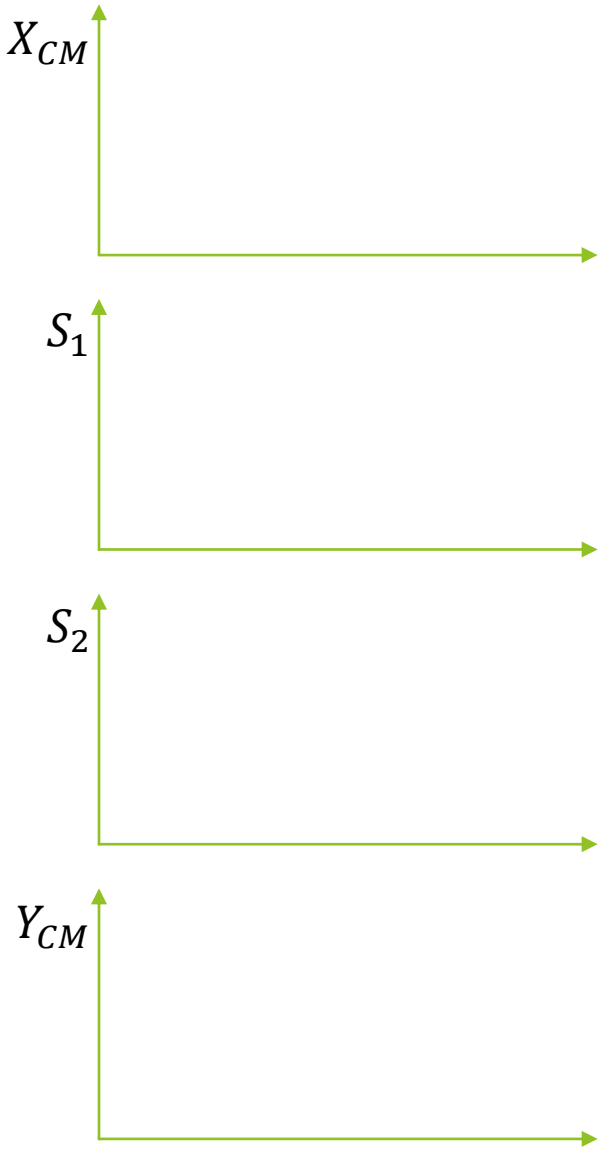


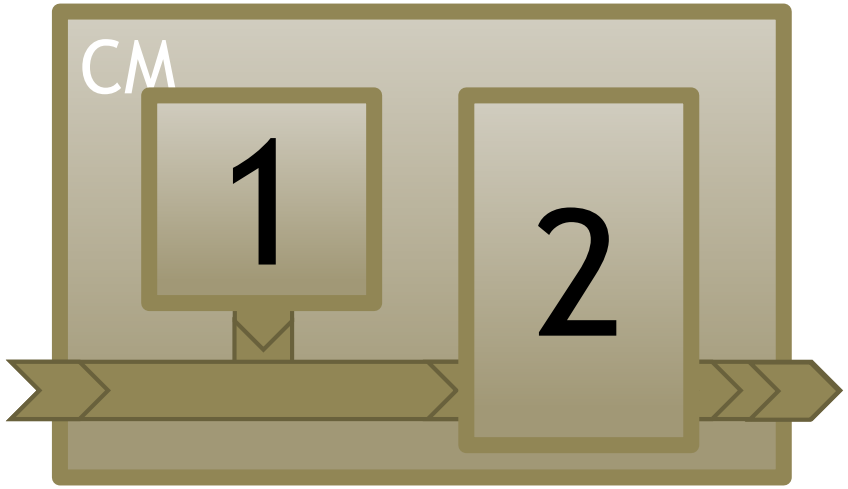
flatten



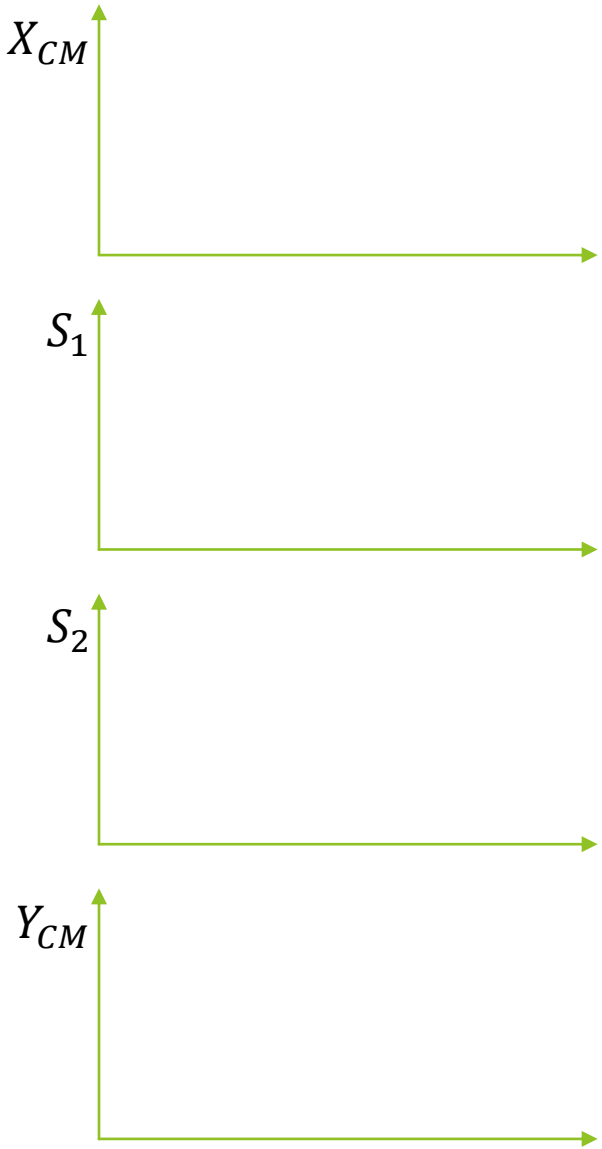
$$CM = \langle X_{self}, Y_{self}, D, MS, IS, ZS \rangle$$

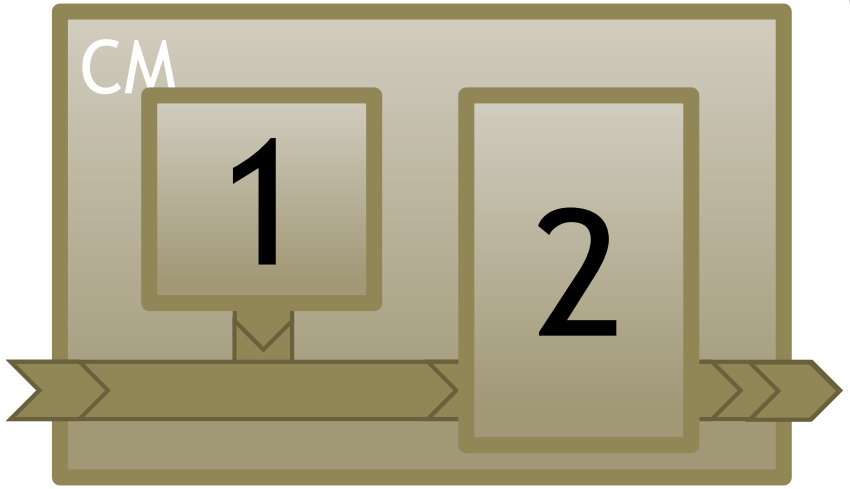
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$





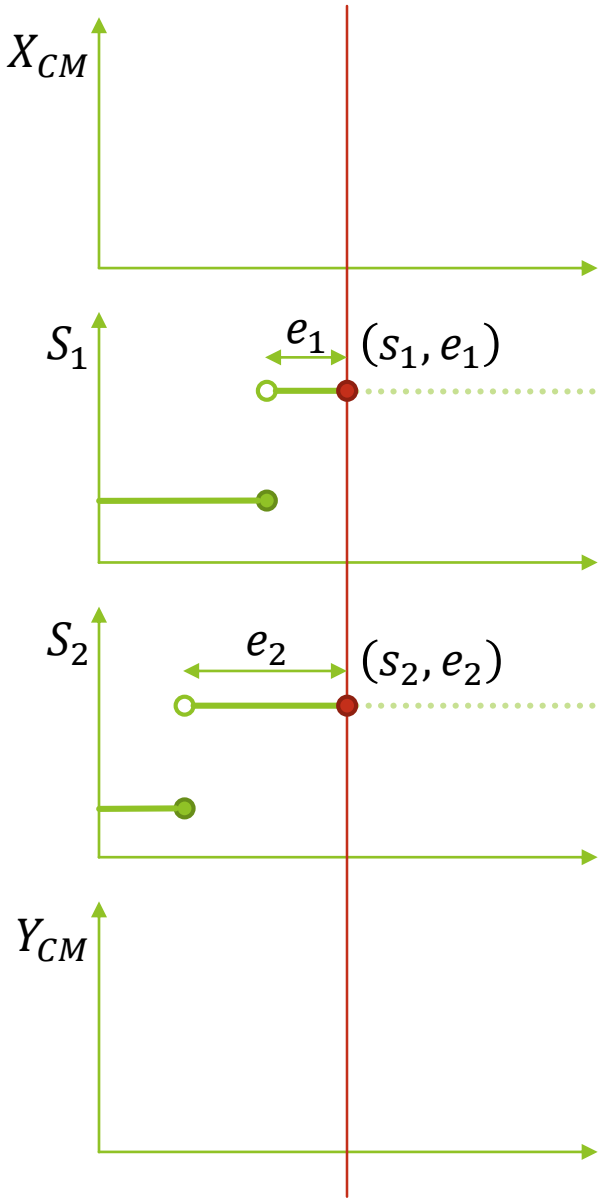
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

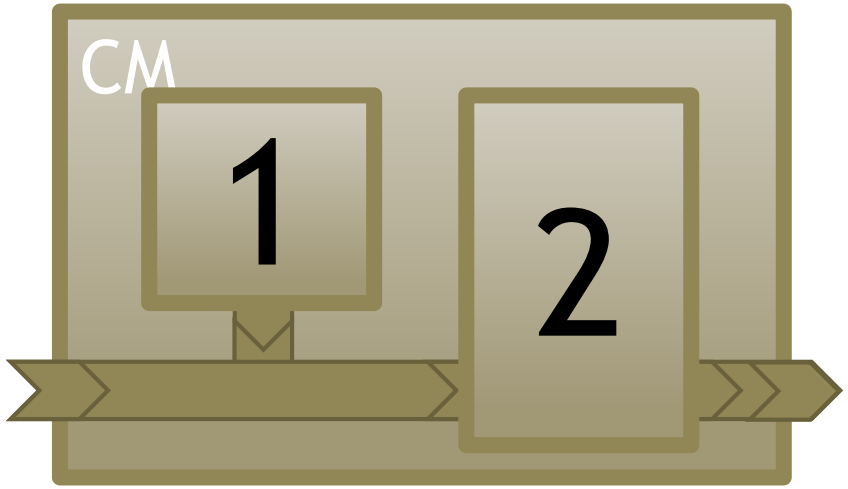




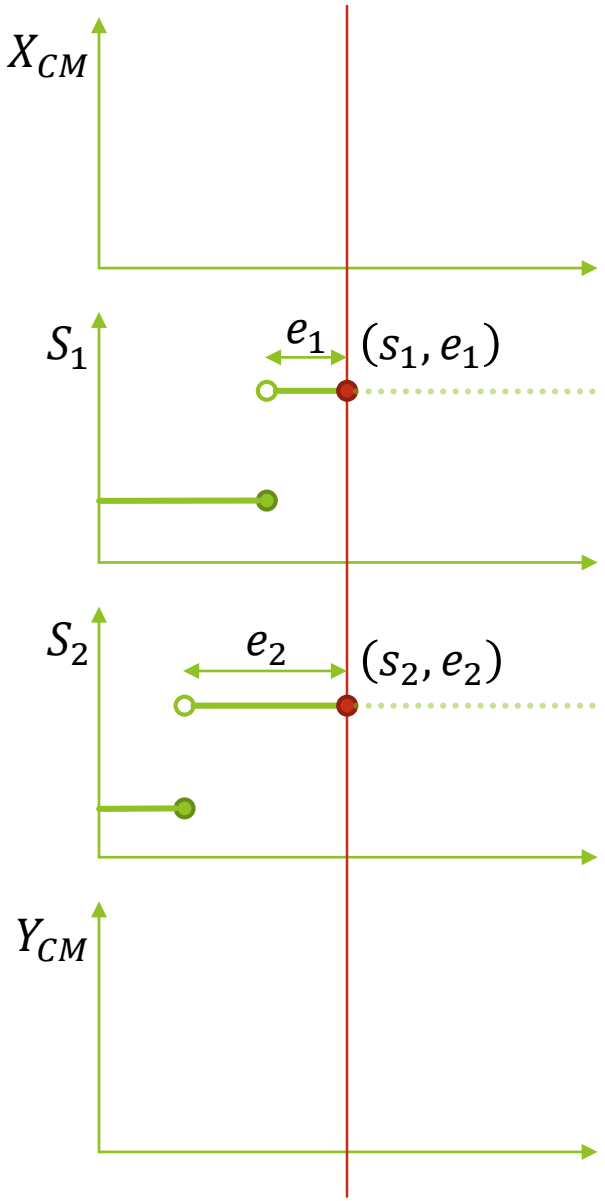
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

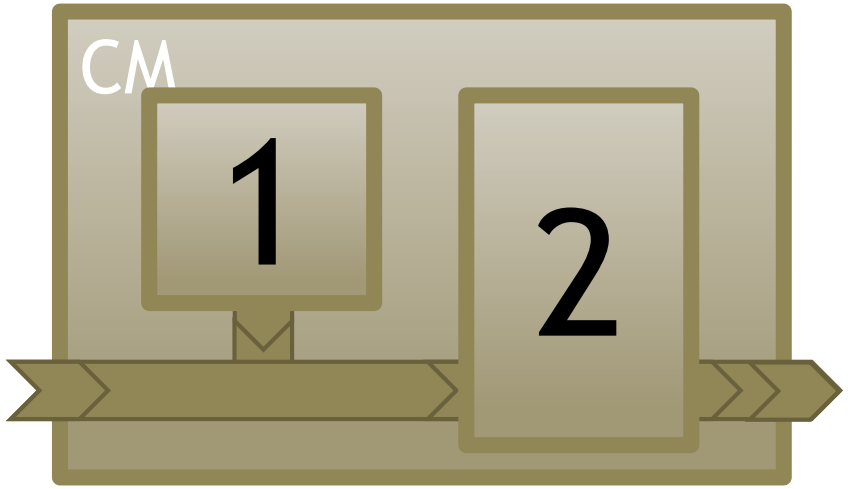
$$X = X_{CM}$$
$$Y = Y_{CM}$$





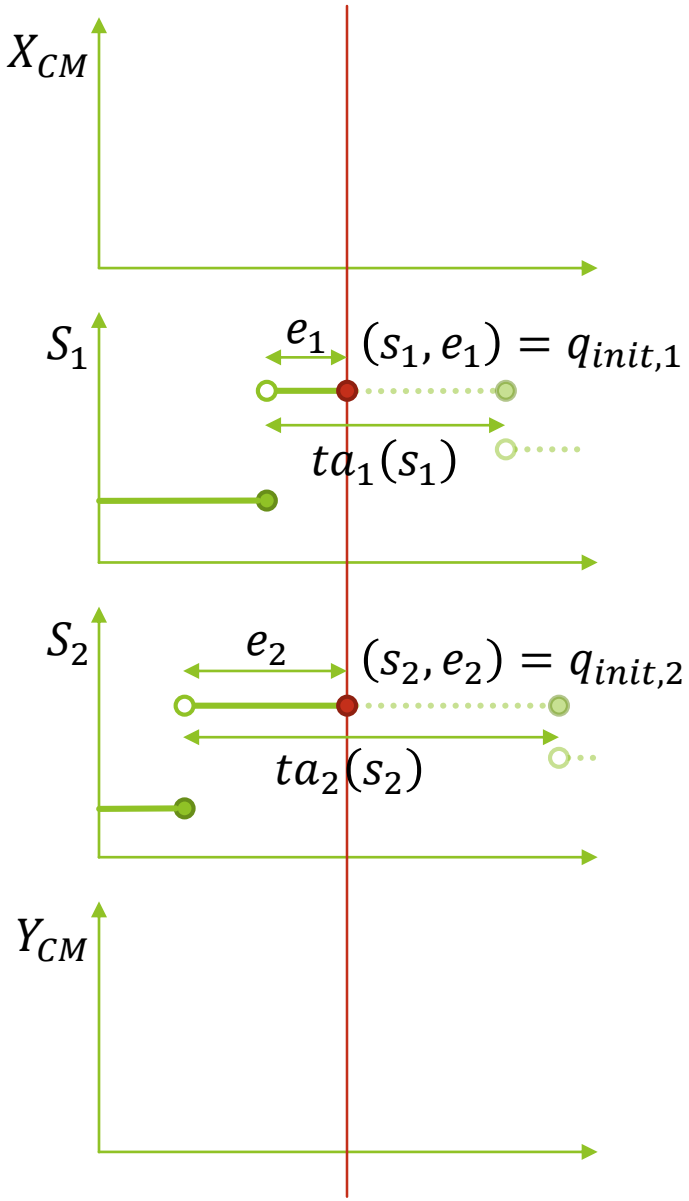
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

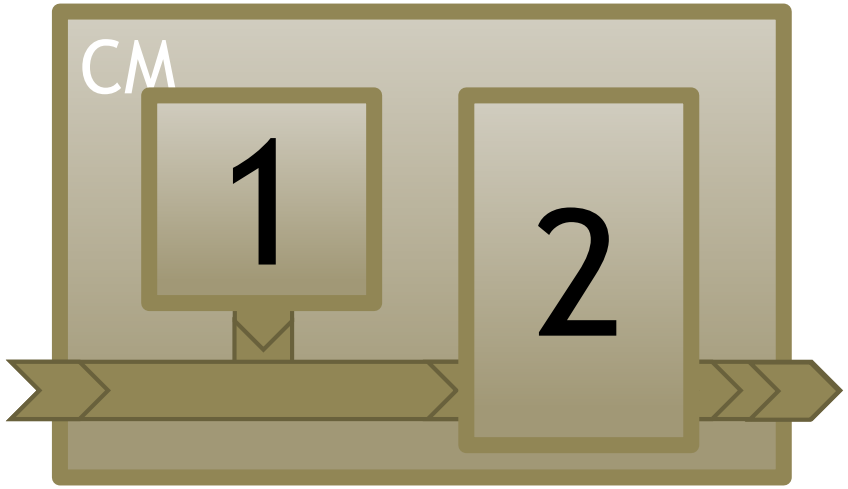




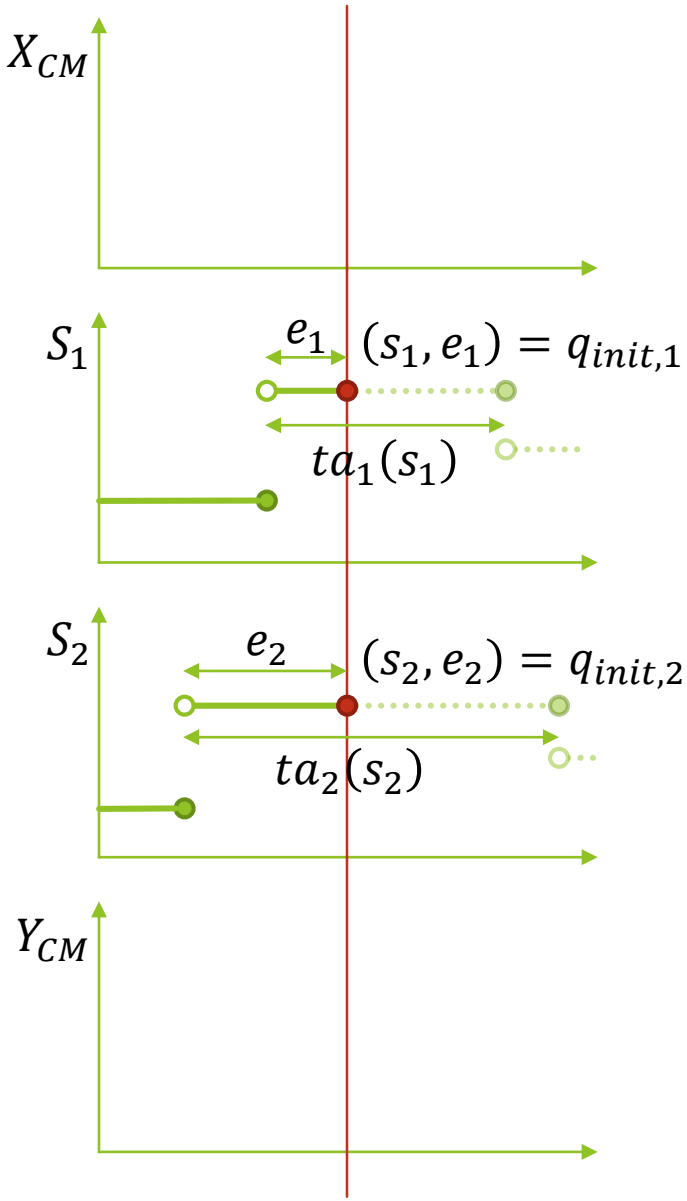
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

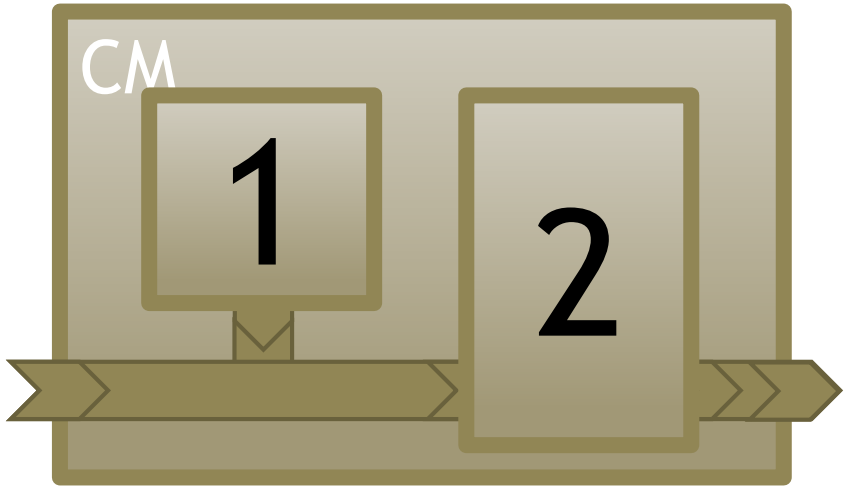
$$S = \times_{i \in D} Q_i$$





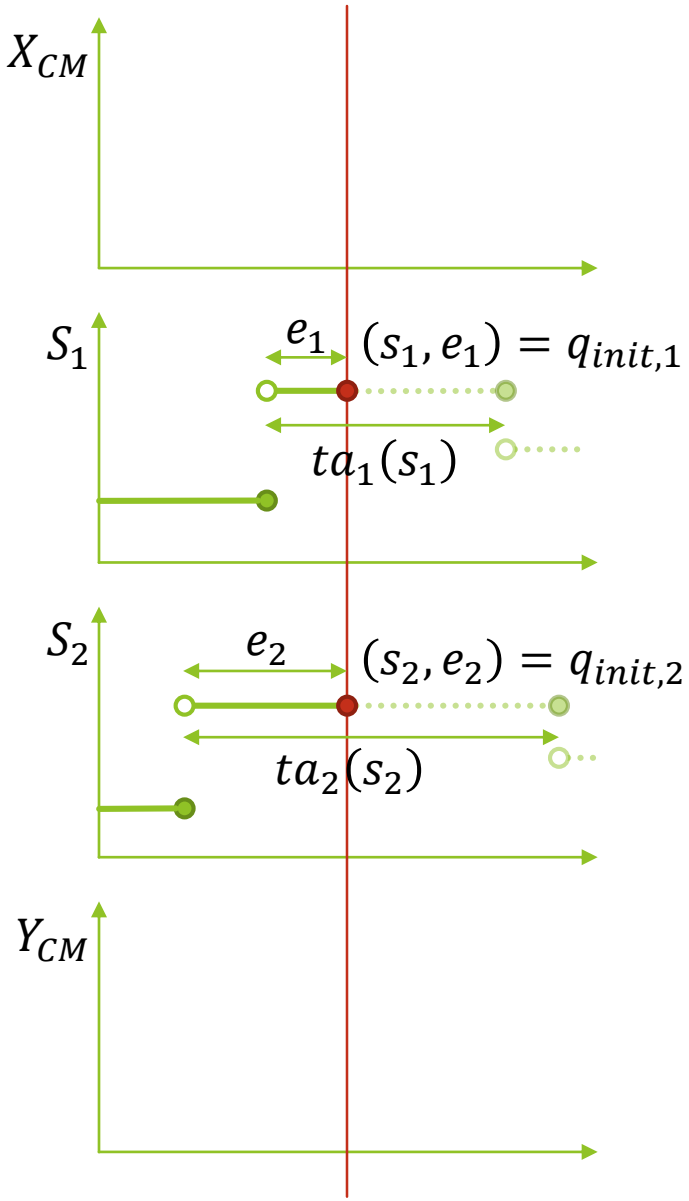
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

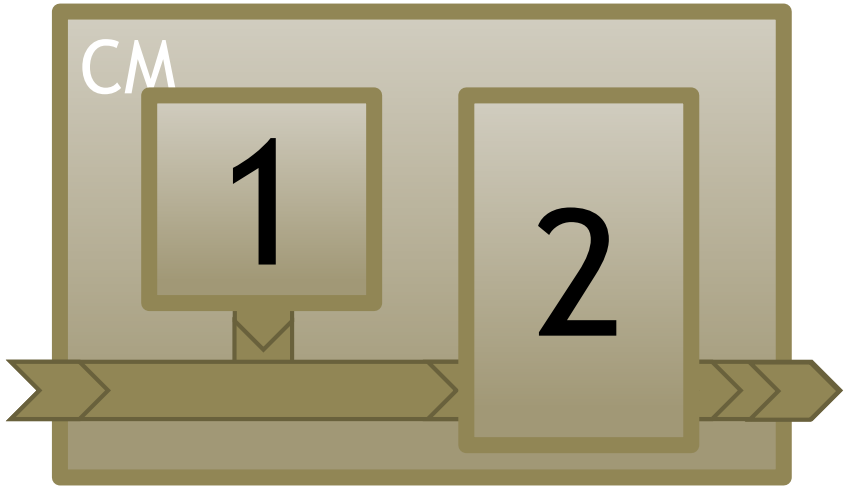




$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$q_{init} = (s_{init}, e_{init})$$

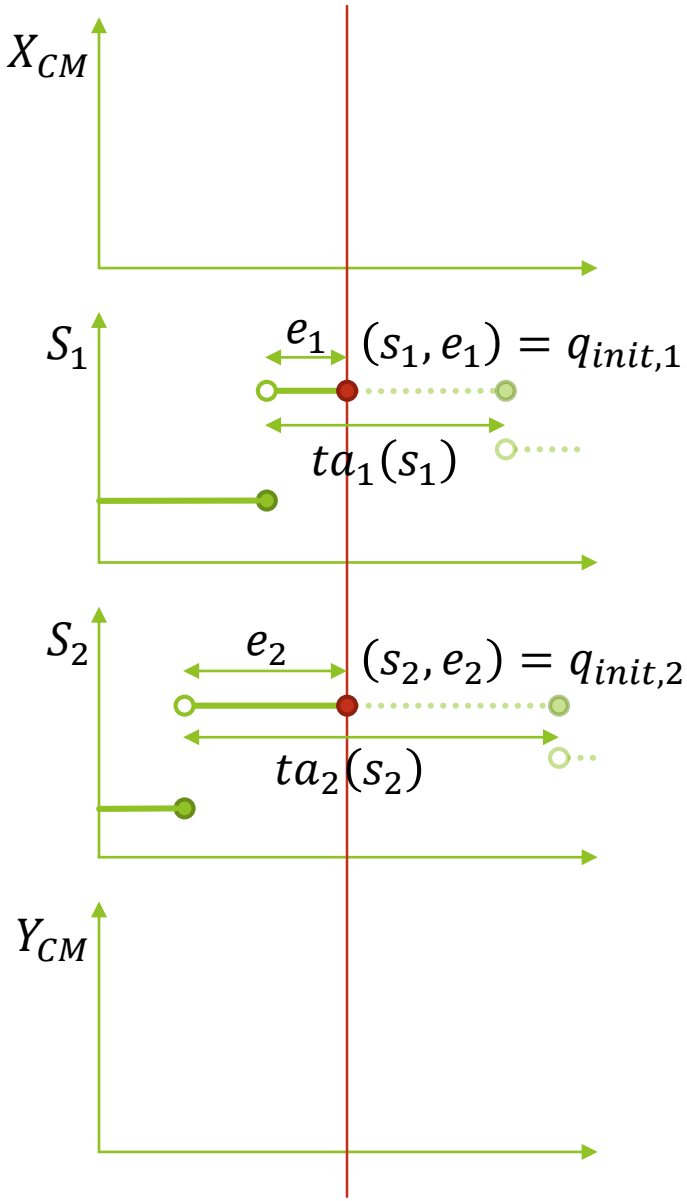


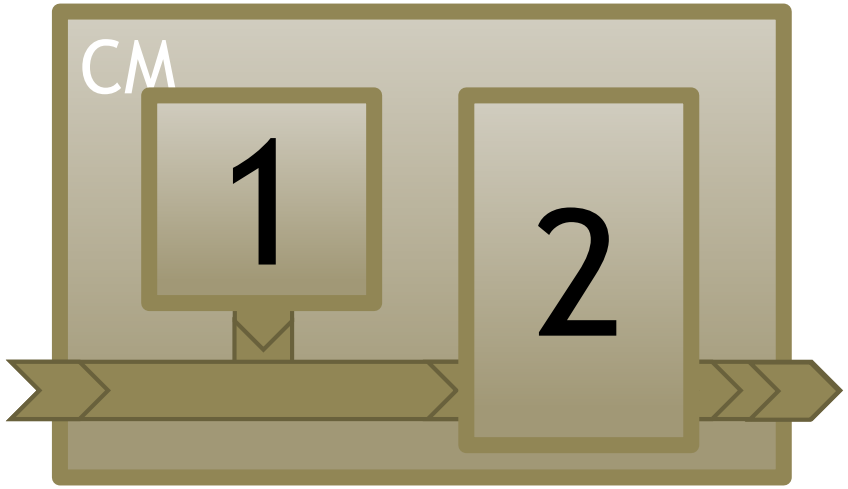


$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$q_{init} = (s_{init}, e_{init})$$

$$s_{init} = \left(\ldots, \left(s_{init,i}, e_{init,i} - e_{init}\right), \ldots\right)$$



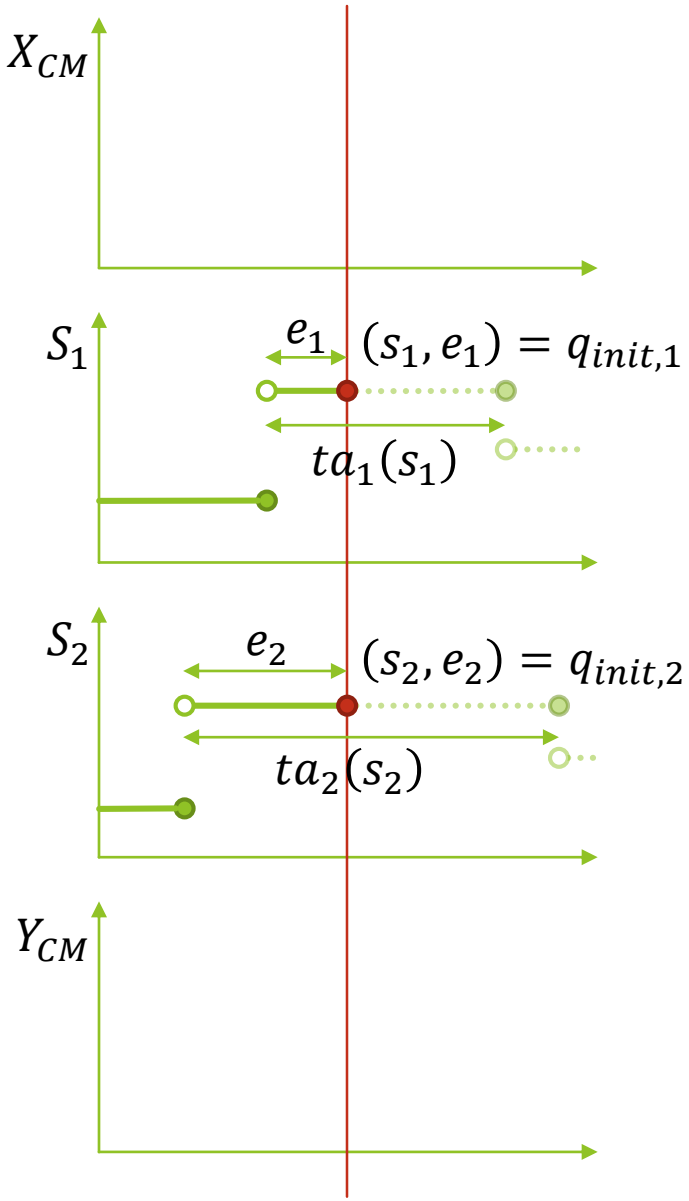


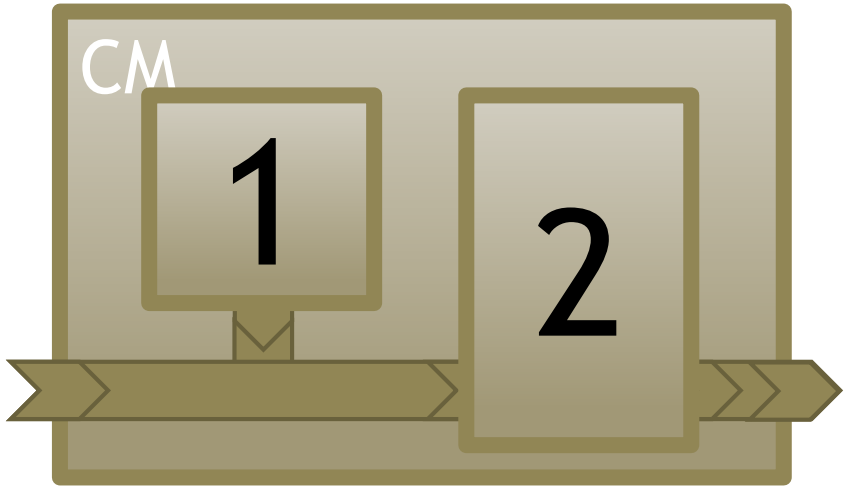
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$q_{init} = (s_{init}, e_{init})$$

$$s_{init} = \left(\dots, \left(s_{init,i}, e_{init,i} - e_{init}\right), \dots\right)$$

$$e_{init} = \min_{i \in D}\{e_{init,i}\}$$





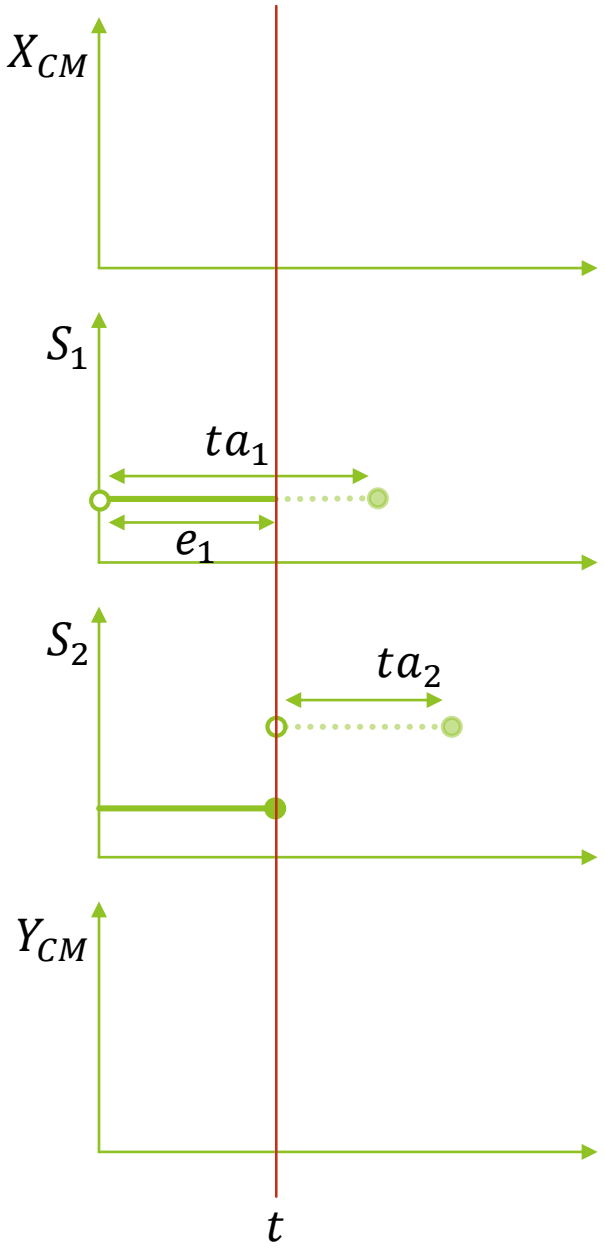
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

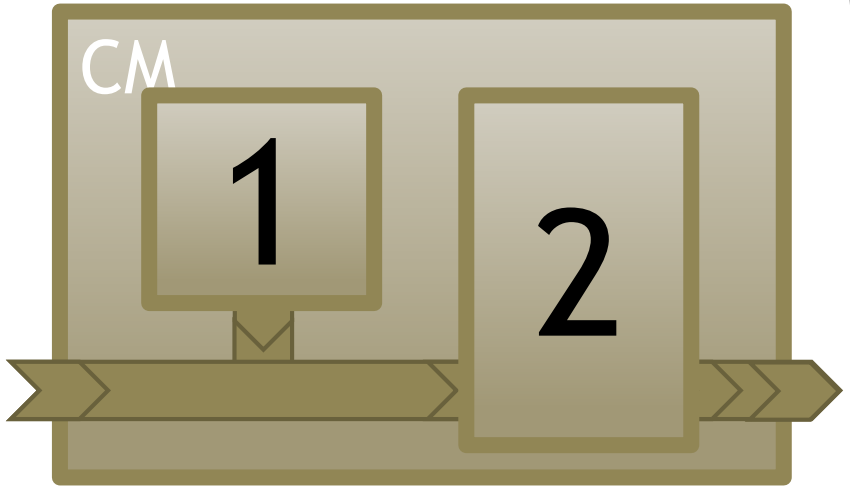
$$q_{init} = (s_{init}, e_{init})$$

$$s_{init} = \left(\dots, \left(s_{init,i}, e_{init,i} - e_{init}\right), \dots\right)$$

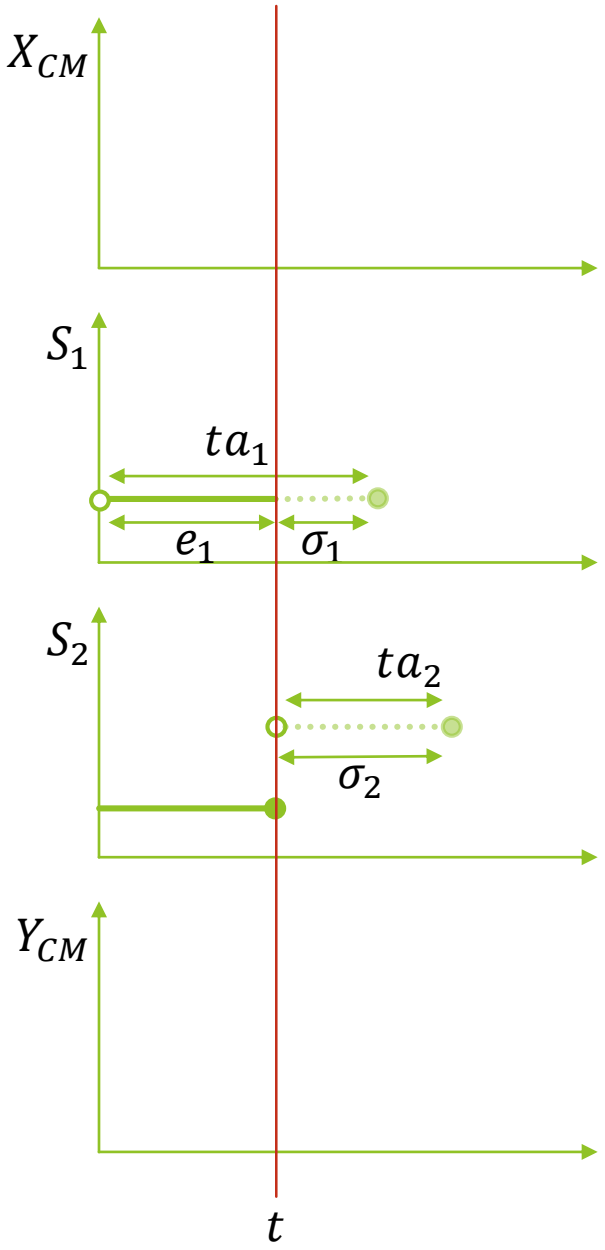
$$e_{init} = \min_{i \in D}\{e_{init,i}\}$$

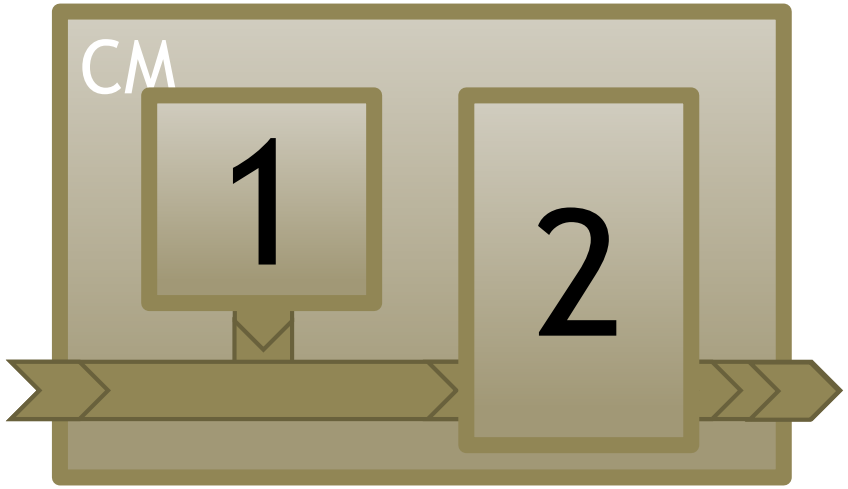
$$\left(s_{init,i}, e_{init,i}\right) = q_{init,i}$$





$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

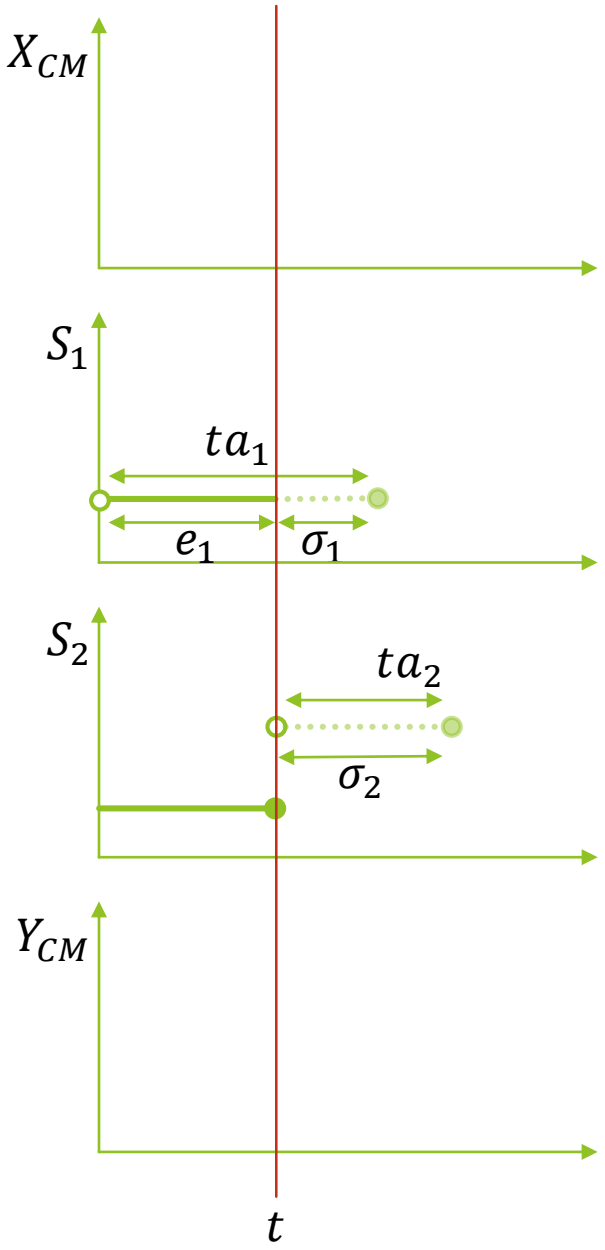


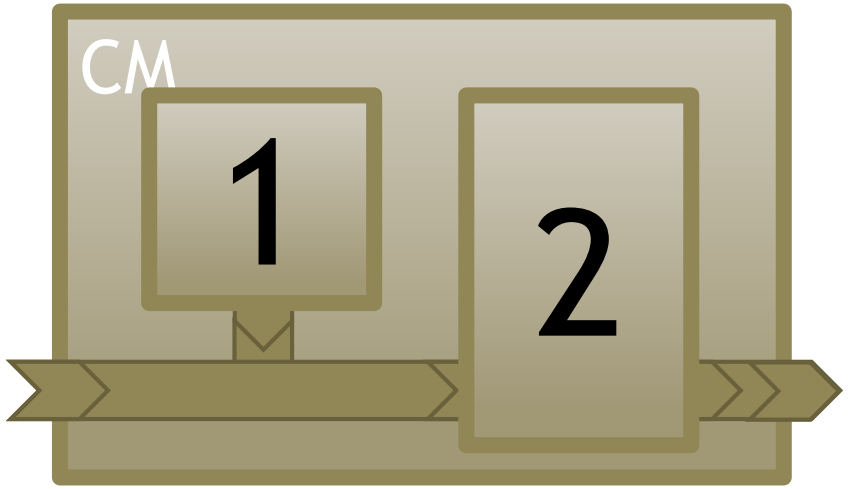


$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$ta : S \rightarrow \mathbb{R}^{+}_{0,+\infty}$$

$$ta(s) = \min_{i \in D}\{\sigma_i = ta_i(s_i) - e_i\}$$





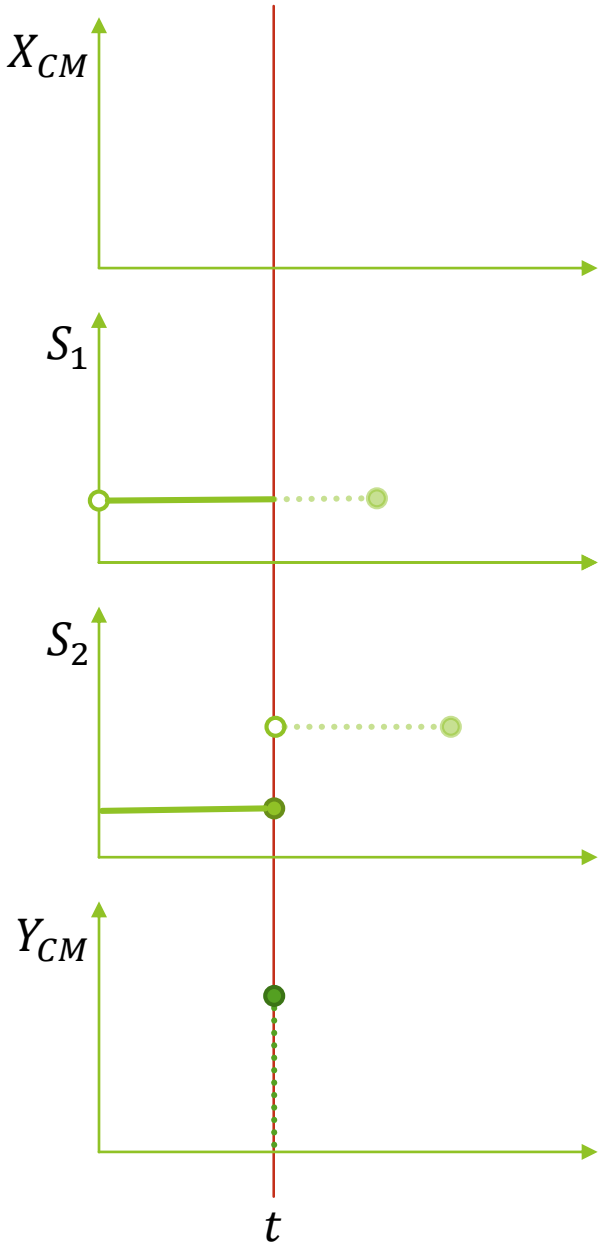
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

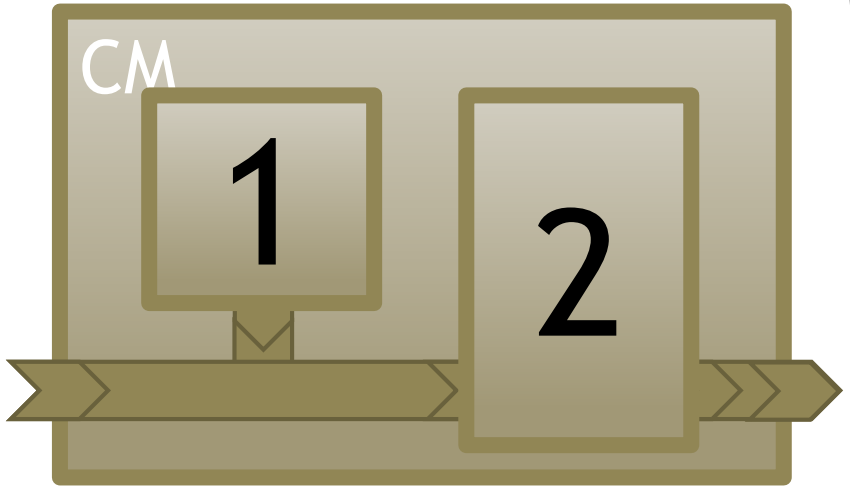
$$ta : S \rightarrow \mathbb{R}^{+}_{0,+\infty}$$

$$ta(s) = \min_{i \in D}\{\sigma_i = ta_i(s_i) - e_i\}$$

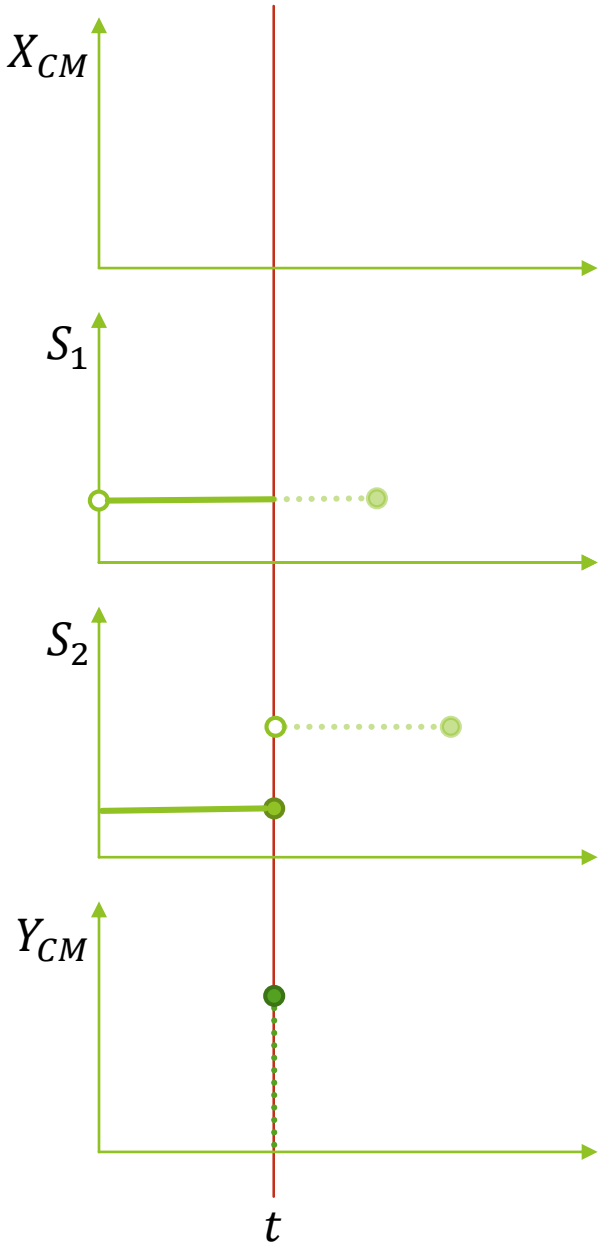
$$IMM(s) = \{i \in D | \sigma_i = ta(s)\}$$

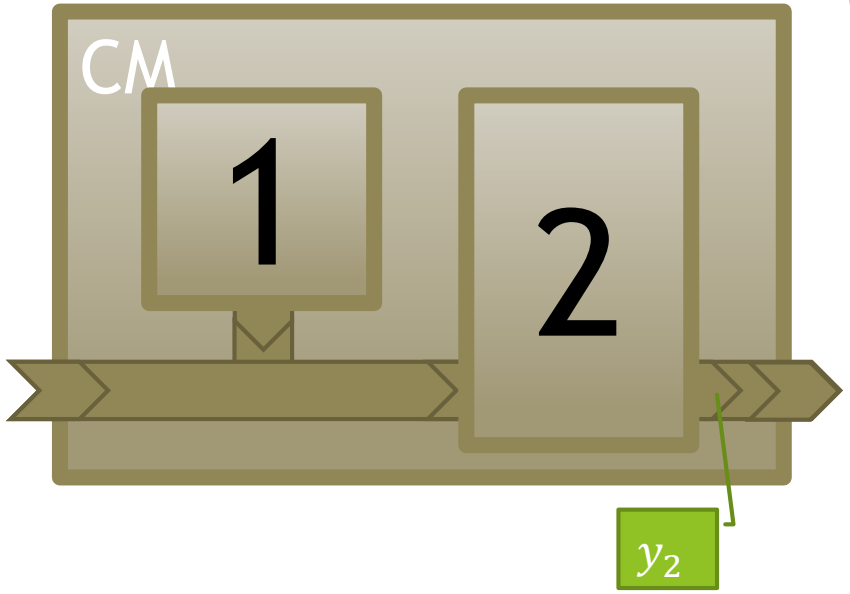
$$select(IMM(s)) = i^*$$





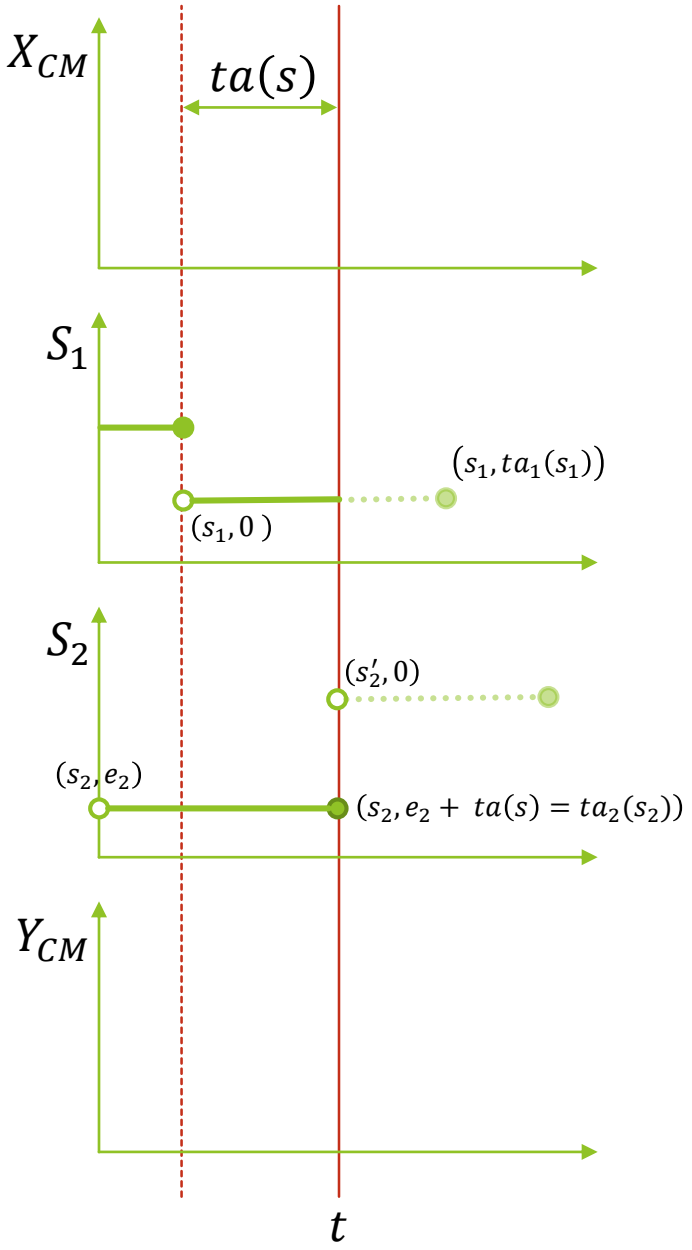
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

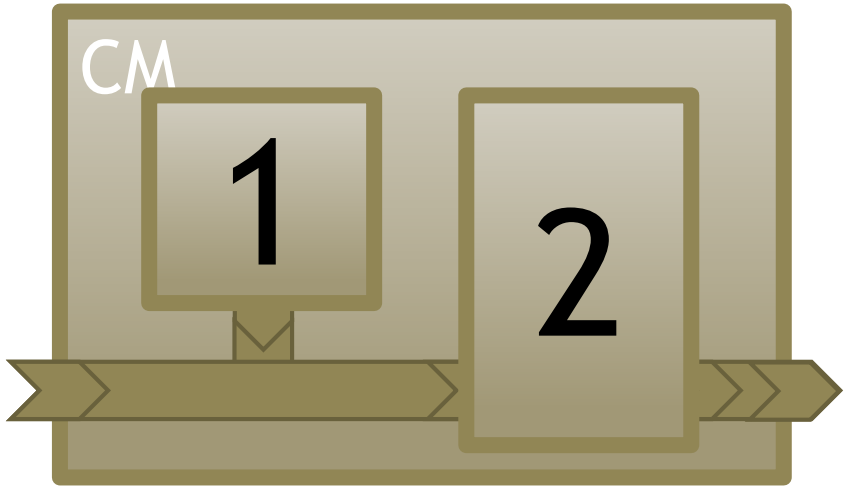




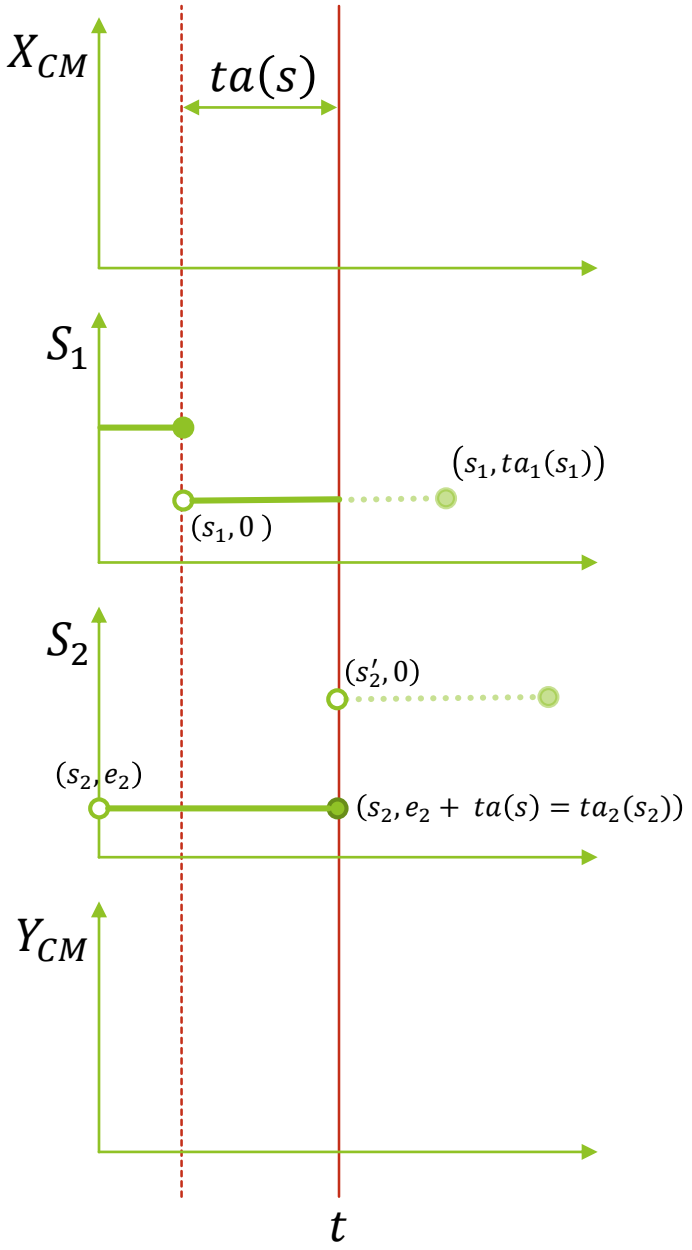
$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

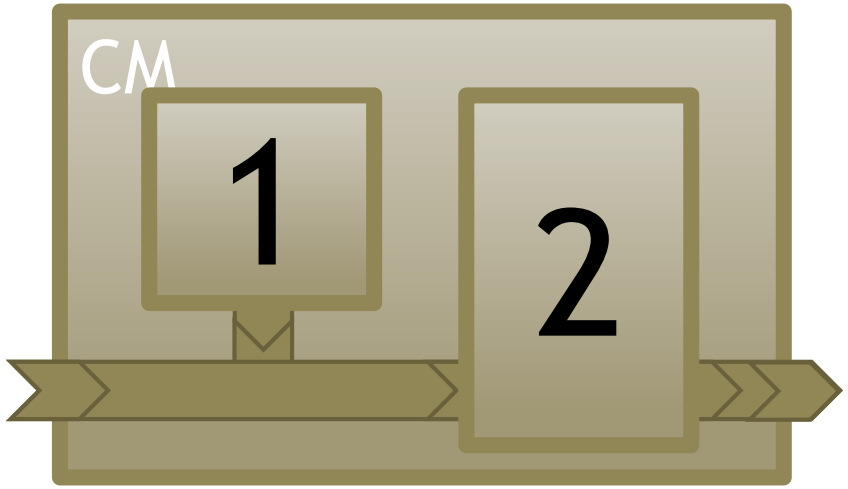
$$\lambda(s) = \begin{cases} Z_{i^*,self}\left(\lambda_{i^*}(s_{i^*})\right) & if\ self \in I_{i^*} \\ \emptyset & if\ self \notin I_{i^*} \end{cases}$$





$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

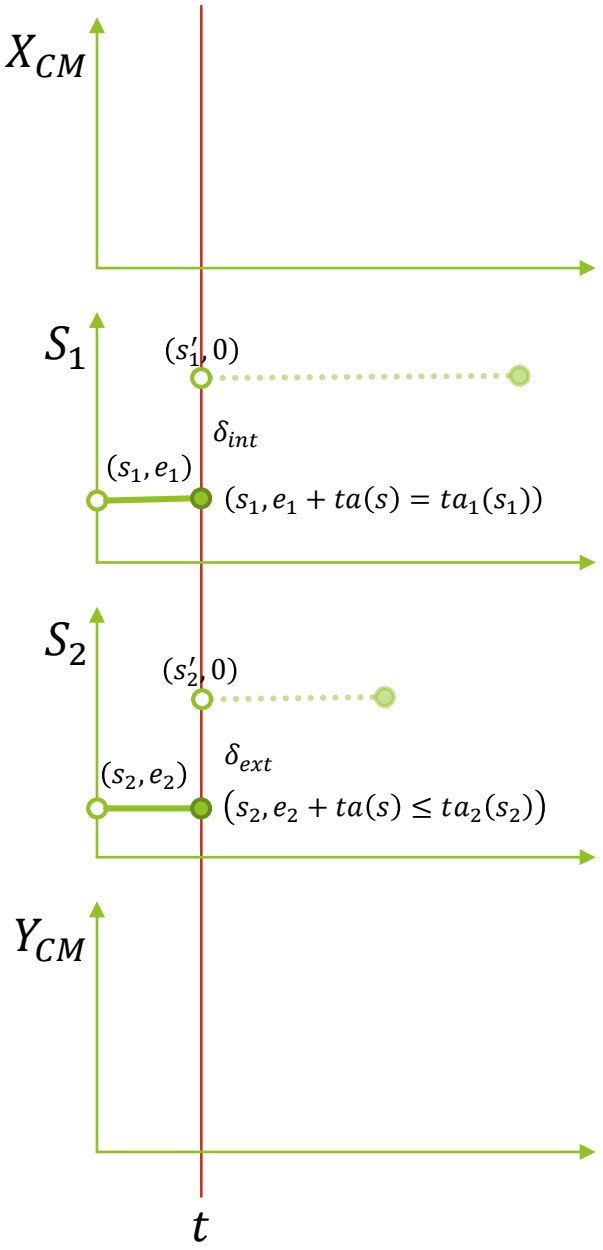


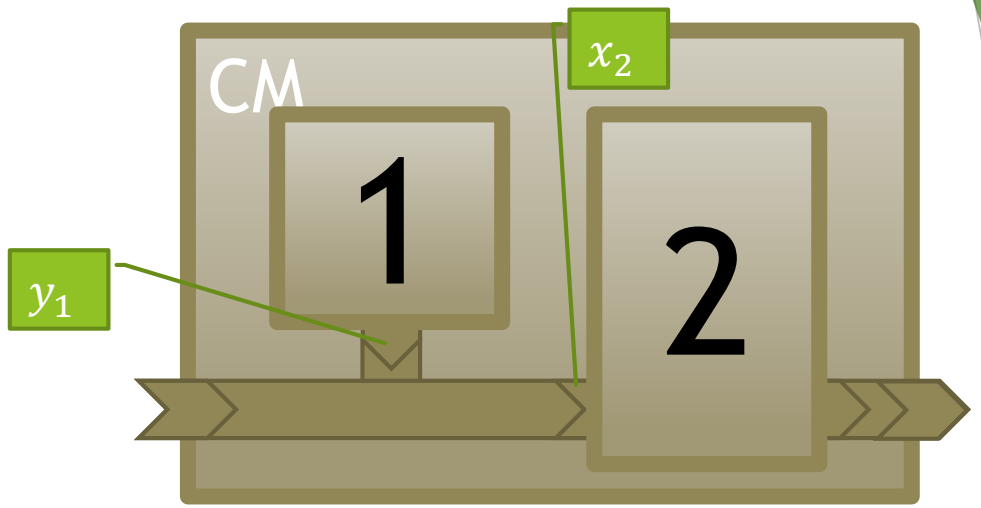


$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$\delta_{int}(s) = \left(\dots, \left(s_j', e_j'\right), \dots\right)$$

$$\left(s_j', e_j'\right) = \begin{cases} \left(\delta_{int,j}\left(s_j\right), 0\right) & for\ j = i^* \\ ? & for\ j \in I_{i^*} \setminus \{self\} \\ \left(s_j, e_j + ta(s)\right) & else \end{cases}$$

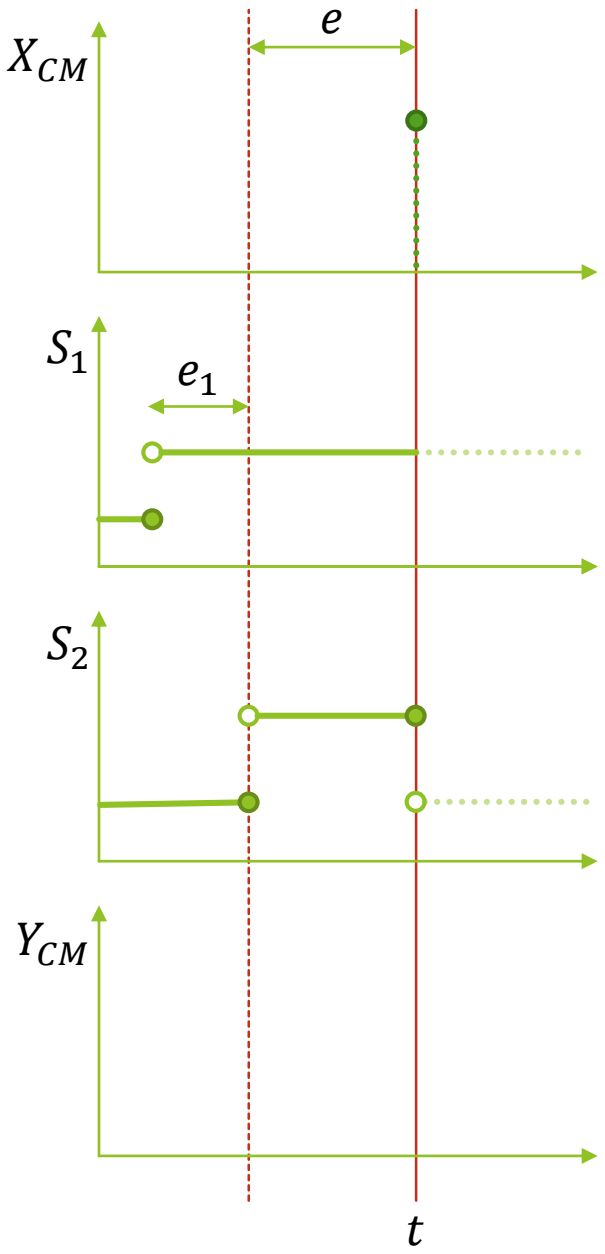


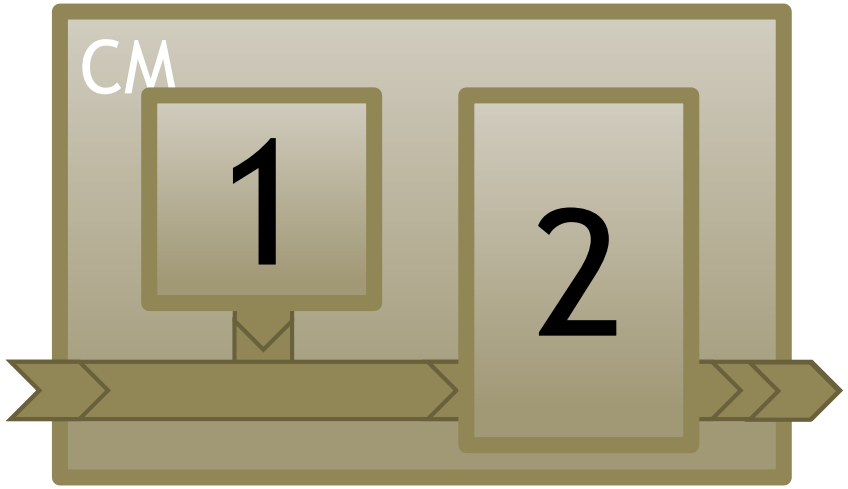


$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

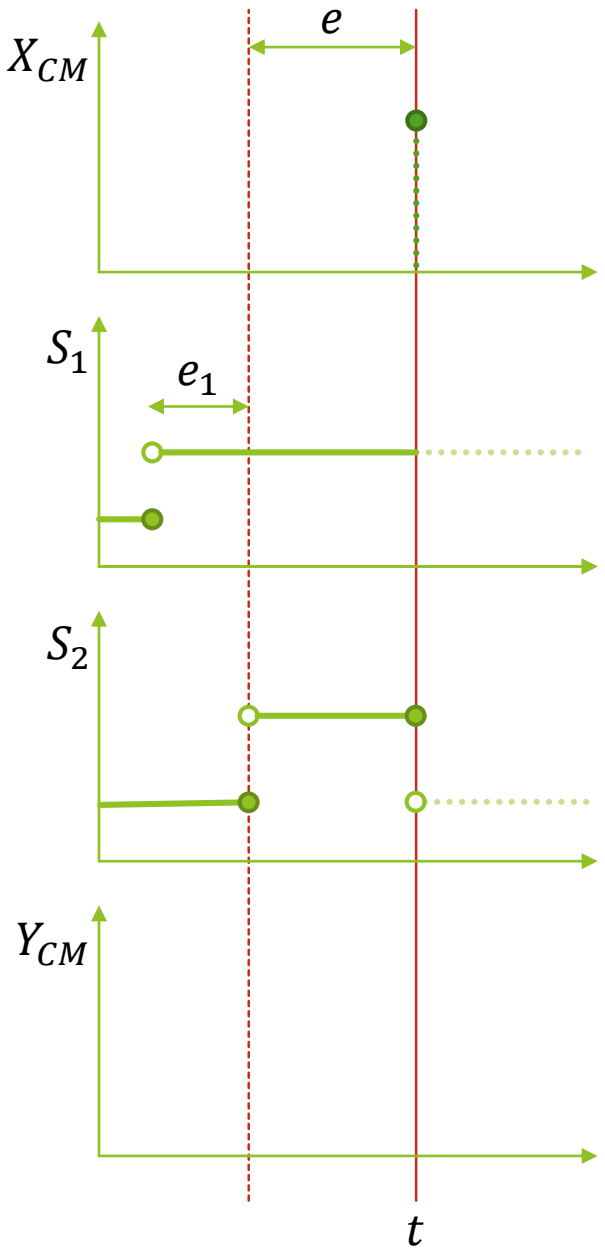
$$\delta_{int}(s) = \left(\dots, \left(s_j', e_j'\right), \dots\right)$$

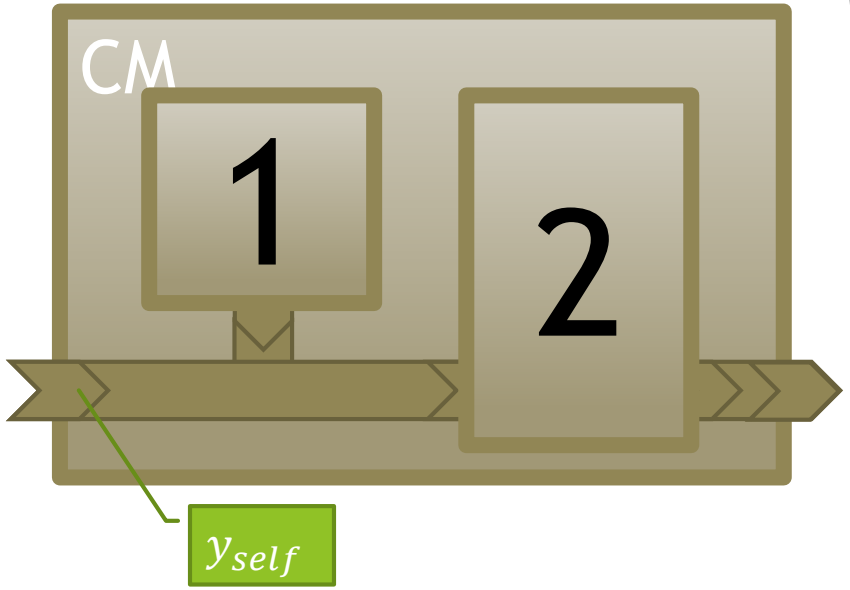
$$\left(s_j', e_j'\right) = \begin{cases} \left(\delta_{int,j}\left(s_j\right), 0\right) & for\ j = i^* \\ \left(\delta_{ext,j}\left(\left(s_j, e_j + ta(s)\right), Z_{i^*,j}\left(\lambda_{i^*}(s_{i^*})\right)\right), 0\right) & for\ j \in I_{i^*} \setminus \{self\} \\ \left(s_j, e_j + ta(s)\right) & else \end{cases}$$





$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$





$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$\delta_{ext}\big((s,e),x\big) = (\dots,(s_i',e_i'),\dots)$$

$$(s_i',e_i') = \begin{cases} \Big(\delta_{ext,i}\Big((s_i,e_i+e),Z_{self,i}(x)\Big),0\Big) & for\ i \in I_{self} \\ (s_i,e_i+e) & else \end{cases}$$

$$flatten(CM) = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

$$X = X_{CM}$$

$$Y = Y_{CM}$$

$$S = \times_{i \in D} Q_i$$

$$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$$

$$q_{init} = (s_{init}, e_{init}) \in Q$$

$$s_{init} = (\dots, (s_{init,i}, e_{init,i} - e_{init}), \dots)$$

$$e_{init} = \min_{i \in D} \{e_{init,i}\}$$

$$(s_{init,i}, e_{init,i}) = q_{init,i}$$

$$\delta_{int}(s) = (\dots, (s_j', e_j'), \dots)$$

$$(s_j', e_j') = \begin{cases} (\delta_{int,j}(s_j), 0) & for\ j = i^* \\ \left(\delta_{ext,j}\left((s_j, e_j + ta(s)), Z_{i^*,j}(\lambda_{i^*}(s_{i^*}))\right), 0\right) & for\ j \in I_{i^*} \\ (s_j, e_j + ta(s)) & else \end{cases}$$

$$\delta_{ext}((s, e), x) = (\dots, (s_i', e_i'), \dots)$$

$$(s_i', e_i') = \begin{cases} \left(\delta_{ext,i}\left((s_i, e_i + e), Z_{self,i}(x)\right), 0\right) & for\ i \in I_{self} \\ (s_i, e_i + e) & else \end{cases}$$

$$\lambda(s) = \begin{cases} Z_{i^*,self}(\lambda_{i^*}(s_{i^*})) & if\ self \in I_{i^*} \\ \phi & if\ self \notin I_{i^*} \end{cases}$$

$$i^* = select(IMM(s))$$

$$IMM(s) = \{i \in D | \sigma_i = ta(s)\}$$

$$ta(s) = \min_{i \in D} \{\sigma_i = ta_i(s_i) - e_i\}$$

Applications of DEVS

DEVS in practice

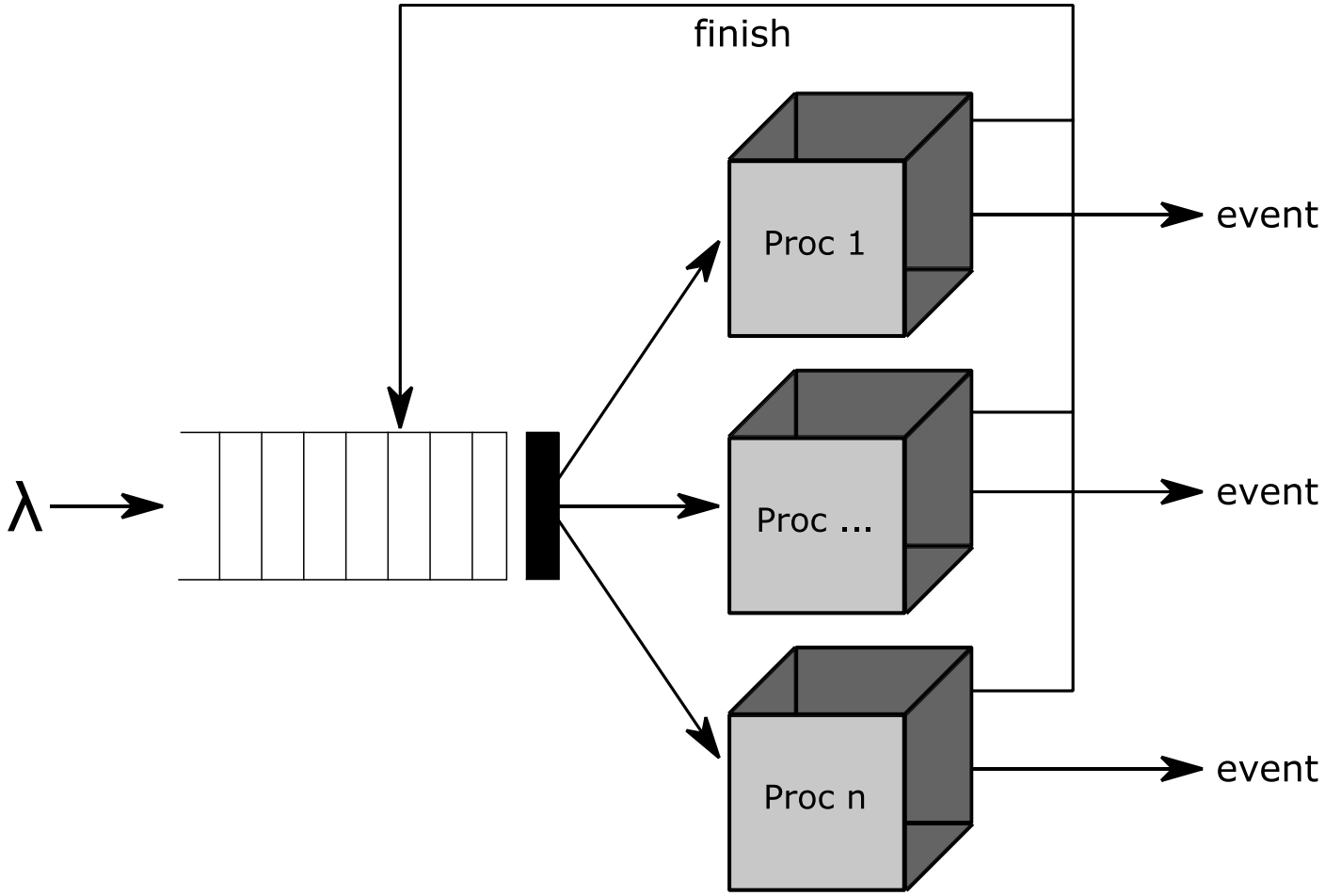

finish
λ
Proc 1
Proc ...
Proc n
event
event
event

Table 3. M/M/1 Queueing System

$$\rho = \lambda W_s, \quad p_n = P[N = n] = (1-\rho)\rho^n, \quad n = 0, 1, \cdots.$$

$$P[N \geq n] = \rho^n, \quad n = 0, 1, \cdots.$$

$$L = E[N] = \lambda W = \frac{\rho}{1-\rho}, \quad \sigma_N^2 = \frac{\rho}{(1-\rho)^2}.$$

$$L_q = \lambda W_q = \frac{\rho^2}{1-\rho}, \quad \sigma_{N_q}^2 = \frac{\rho^2(1+\rho-\rho^2)}{(1-\rho)^2}.$$

$$E[N_q|N_q > 0] = \frac{1}{1-\rho}, \quad Var[N_q|N_q > 0] = \frac{\rho}{(1-\rho)^2}.$$

$$W[t] = P[w \leq t] = 1 - \exp\left(\frac{-t}{W}\right), \quad P[w > t] = \exp\left(\frac{-t}{W}\right).$$

$$W = E[w] = \frac{W_s}{1-\rho}, \quad \sigma_w^2 = W^2.$$

$$\pi_w[r] = W \ln\left(\frac{100}{100-r}\right), \quad \pi_w[90] = W \ln 10, \quad \pi_w[95] = W \ln 20$$

$$W_q[t] = P[q \leq t] = 1 - \rho\exp\left(\frac{-t}{W}\right), \quad P[q > t] = \rho\exp\left(\frac{-t}{W}\right).$$

$$W_q = \frac{\rho W_S}{1-\rho}, \quad \sigma_q^2 = \frac{(2-\rho)\rho W_s^2}{(1-\rho)^2}.$$

$$\pi_q[r] = \max\left\{W \ln\left(\frac{100\rho}{100-r}\right), 0\right\}.$$

$$\pi_q[90] = \max\{W \ln(10\rho), 0\}, \quad \pi_q[95] = \max\{W \ln(20\rho), 0\}.$$

Table 4. M/M/1/K Queueing System

$$p_n = \begin{cases} \dfrac{(1-a)a^n}{(1-a^{K+1})} & \text{if } \lambda \neq \mu, \\ \dfrac{1}{K+1} & \text{if } \lambda = \mu, \end{cases}$$

$n = 0, 1, \cdots, K$, where $a = \lambda W_s$.

$\lambda_a = (1-p_K)\lambda$, Mean arrival rate into system.

$$L = \begin{cases} \dfrac{a[1-(K+1)a^K + Ka^{K+1}]}{(1-a)(1-a^{K+1})} & \text{if } \lambda \neq \mu, \\ \dfrac{K}{2} & \text{if } \lambda = \mu. \end{cases}$$

$$L_q = L - (1-p_0), \quad q_n = \frac{p_n}{1-p_K}, n = 0, 1, \cdots, K-1.$$

$$W[t] = 1 - \sum_{n=0}^{K-1} q_n Q[n; \mu t],$$

where

$$q[n; \mu t] = e^{-\mu t} \sum_{k=0}^{n} \frac{\mu t}{k!}.$$

$$W = \frac{L}{\lambda_a}, \quad W_q = \frac{L_q}{\lambda_a}.$$

$$W_q[t] = 1 - \sum_{n=0}^{K-2} q_{n+1} Q[n; \mu t].$$

$$E[q|q > 0] = \frac{W_q}{1-p_0}, \quad \rho = (1-p_K)a.$$

http://irh.inf.unideb.hu/user/jsztrik/education/09/english/3f.html

Table 3. M/M/1 Queueing System

$$\rho = \lambda W_s, \quad p_n = P[N = n] = (1-\rho)\rho^n, \quad n = 0, 1, \cdots.$$

$$P[N \geq n] = \rho^n, \quad n = 0, 1, \cdots.$$

$$L = E[N] = \lambda W = \frac{\rho}{1-\rho}, \quad \sigma_N^2 = \frac{\rho}{(1-\rho)^2}.$$

$$L_q = \lambda W_q = \frac{\rho^2}{1-\rho}, \quad \sigma_{N_q}^2 = \frac{\rho^2(1+\rho-\rho^2)}{(1-\rho)^2}.$$

$$E[N_q | N_q > 0] = \frac{1}{1-\rho}, \quad Var[N_q | N_q > 0] = \frac{\rho}{(1-\rho)^2}.$$

$$W[t] = P[w \leq t] = 1 - \exp\left(\frac{-t}{W}\right), \quad P[w > t] = \exp\left(\frac{-t}{W}\right).$$

$$W = E[w] = \frac{W_s}{1-\rho}, \quad \sigma_w^2 = \frac{W_s^2(2-\rho)\rho}{(1-\rho)^2}.$$

$$\pi_w[r] = W \ln\left(\frac{100}{100-r}\right), \quad \pi_w[90] = W \ln 10, \quad \pi_w[95] = W \ln 20$$

$$W_q[t] = P[q \leq t] = 1 - \rho \exp\left(\frac{-t}{W}\right), \quad P[q > t] = \rho \exp\left(\frac{-t}{W}\right).$$

$$W_q = \frac{\rho W_S}{1-\rho}, \quad \sigma_q^2 = \frac{(2-\rho)\rho W_s^2}{(1-\rho)^2}.$$

$$\pi_q[r] = \max\left\{W \ln\left(\frac{100\rho}{100-r}\right), 0\right\}.$$

$$\pi_q[90] = \max\{W \ln(10\rho), 0\}, \quad \pi_q[95] = \max\{W \ln(20\rho), 0\}.$$

Table 4. M/M/1/K Queueing System

$$p_n = \begin{cases} \dfrac{(1-a)a^n}{(1-a^{K+1})} & \text{if } \lambda \neq \mu, \\ \dfrac{1}{K+1} & \text{if } \lambda = \mu, \end{cases}$$

$n = 0, 1, \cdots, K$, where $a = \lambda W_s$.

$\lambda_a = (1-p_K)\lambda$, Mean arrival rate into system.

$$L = \begin{cases} \dfrac{a[1-(K+1)a^K + Ka^{K+1}]}{(1-a)(1-a^{K+1})} & \text{if } \lambda \neq \mu, \\ \dfrac{K}{2} & \text{if } \lambda = \mu. \end{cases}$$

$$L_q = L - (1-p_0), \quad q_n = \frac{p_n}{1-p_K}, n = 0, 1, \cdots, K-1.$$

$$W[t] = 1 - \sum_{n=0}^{K-1} q_n Q[n; \mu t],$$

where

$$q[n; \mu t] = e^{-\mu t} \sum_{k=0}^{n} \frac{\mu t}{k!}.$$

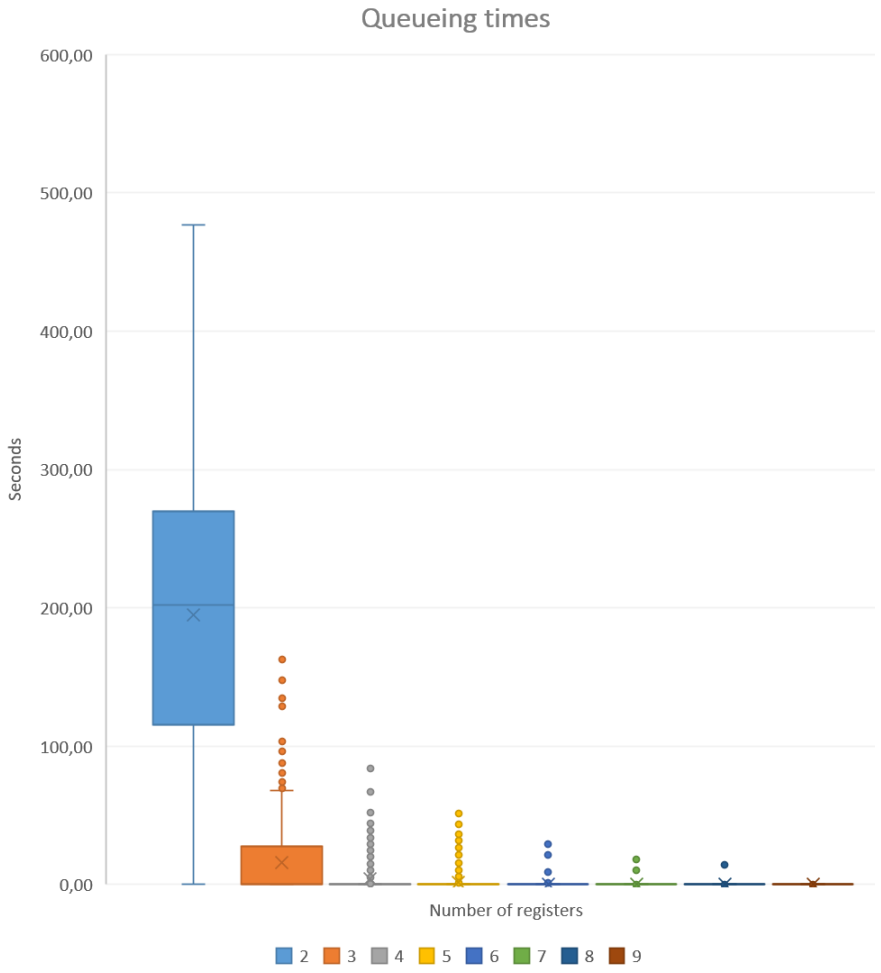
$$W = \frac{L}{\lambda_a}, \quad W_q = \frac{L_q}{\lambda_a}.$$

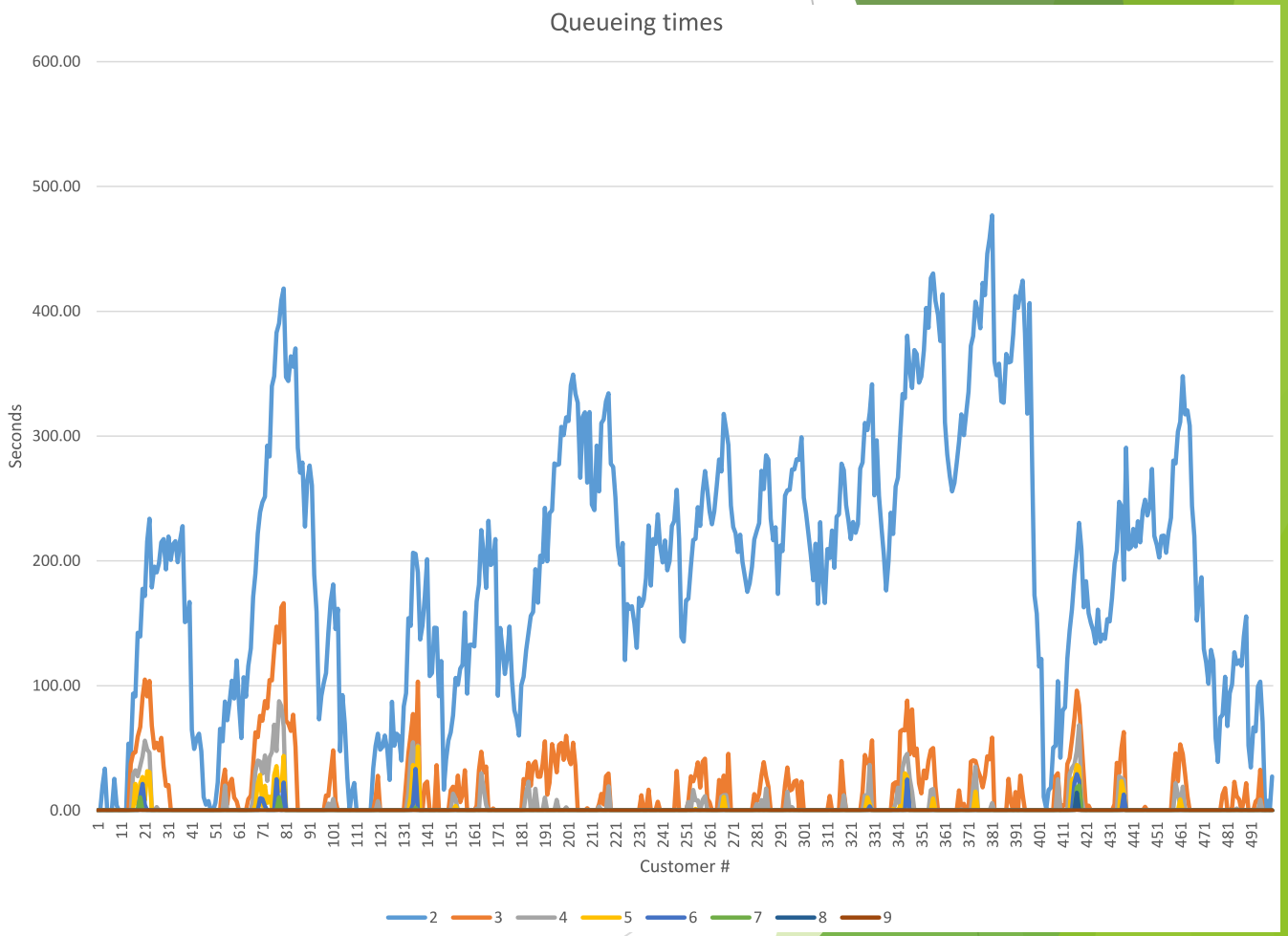
$$W_q[t] = 1 - \sum_{n=0}^{K-2} q_{n+1} Q[n; \mu t].$$

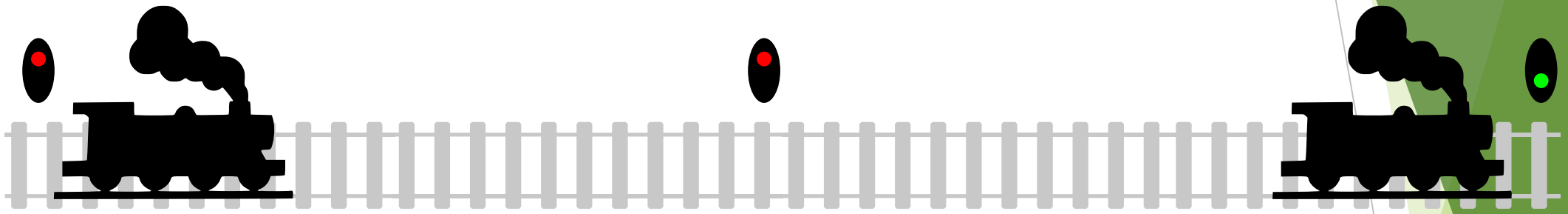
$$E[q | q > 0] = \frac{W_q}{1-p_0}, \quad \rho = (1-p_K)a.$$

16 more types of queueing systems

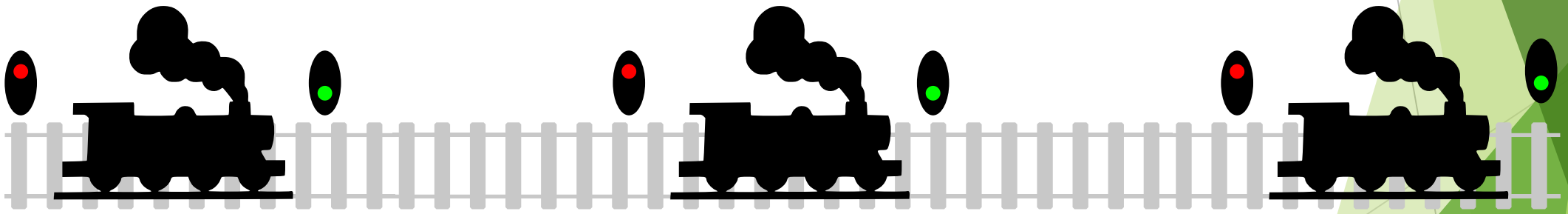
http://irh.inf.unideb.hu/user/jsztrik/education/09/english/3f.html


Queueing times
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Seconds
Number of registers
2
3
4
5
6
7
8
9

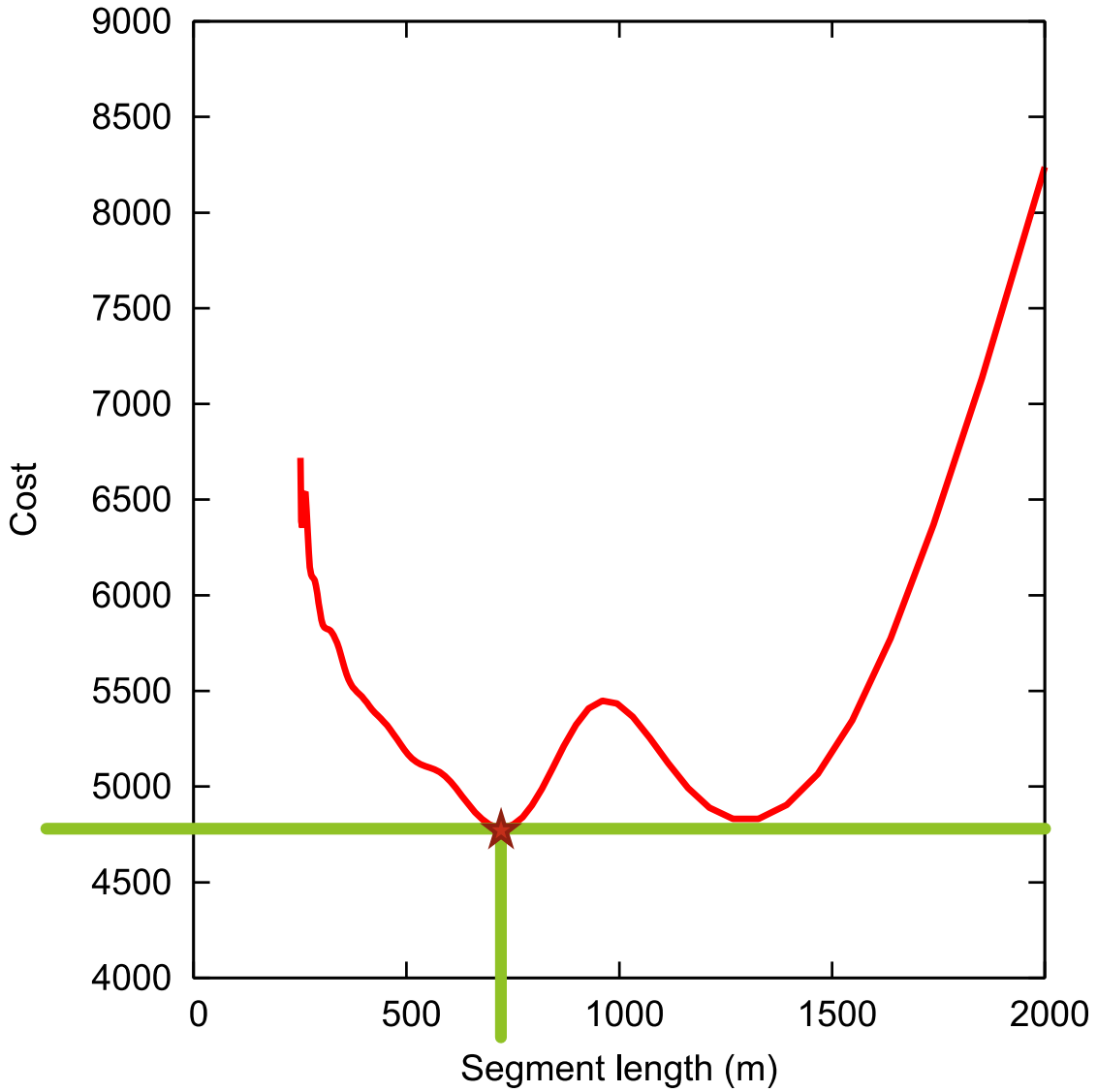

Queueing times
600.00
500.00
400.00
300.00
200.00
100.00
0.00
Seconds
Customer #
2
3
4
5
6
7
8
9

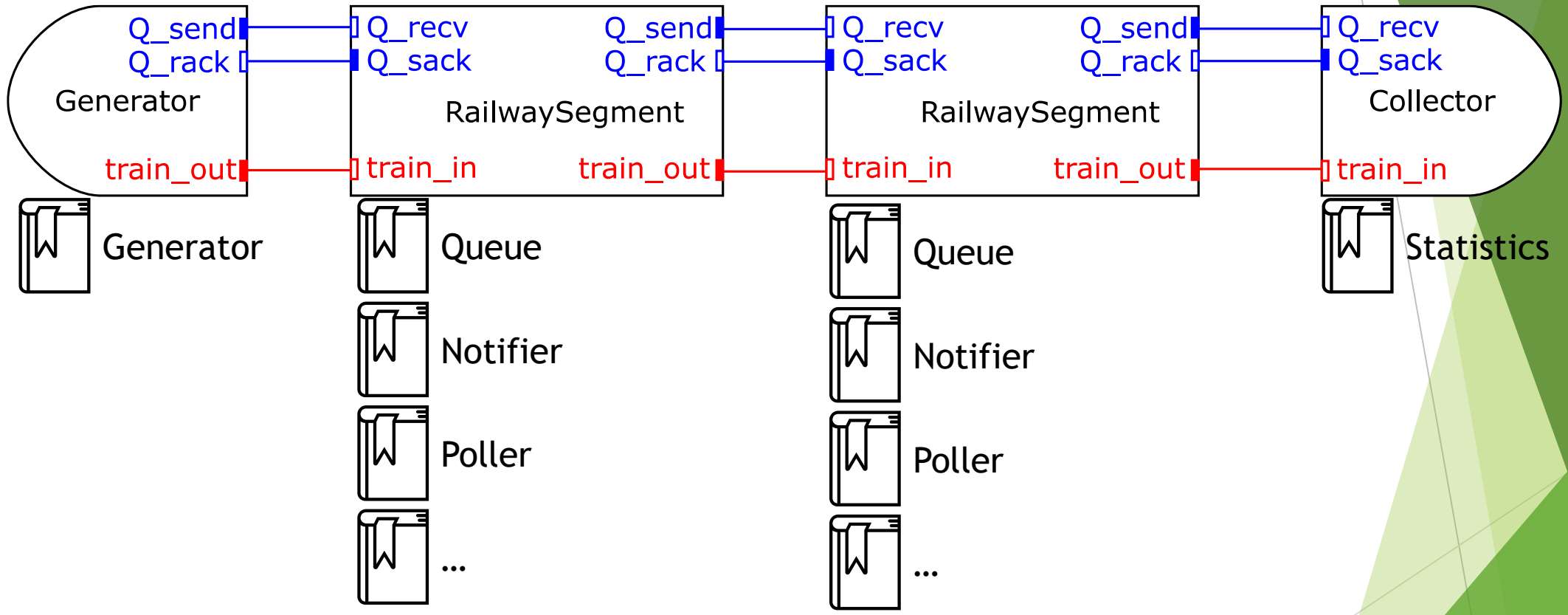


VS.



$$cost = 10 \times \#lights + avg(t_{travel})$$




Q_send
Q_rack
Generator
train_out
Q_recv
Q_sack
Q_send
Q_rack
RailwaySegment
train_in
train_out
Q_recv
Q_sack
Q_send
Q_rack
RailwaySegment
train_in
train_out
Q_recv
Q_sack
Collector
train_in
Generator
Queue
Notifier
Poller
...
Queue
Notifier
Poller
...
Statistics

Custom Atomic DEVS Models

Extending a DEVS repository

Atomic DEVS Specification

$$M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

X : *set of input events*

Y : *set of output events*

S : *set of sequential states*

$q_{init} : Q$

$$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$$

$\delta_{int} : S \to S$

$\delta_{ext} : Q \times X \to S$

$\lambda : S \to Y \cup \{\phi\}$

$ta : S \to \mathbb{R}^{+}_{0,+\infty}$

Atomic DEVS Specification

$$M = \langle X, Y, S, q_{init}, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$$

X : set of input events

Y : set of output

S : sequential states

PATTERNS!

$$= \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$$

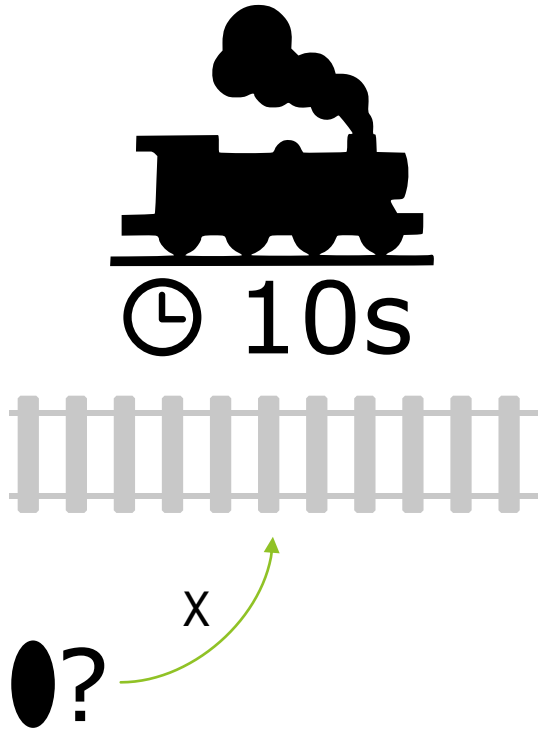
$$\delta_{int} : S \to S$$

$$\delta_{ext} : Q \times X \to S$$

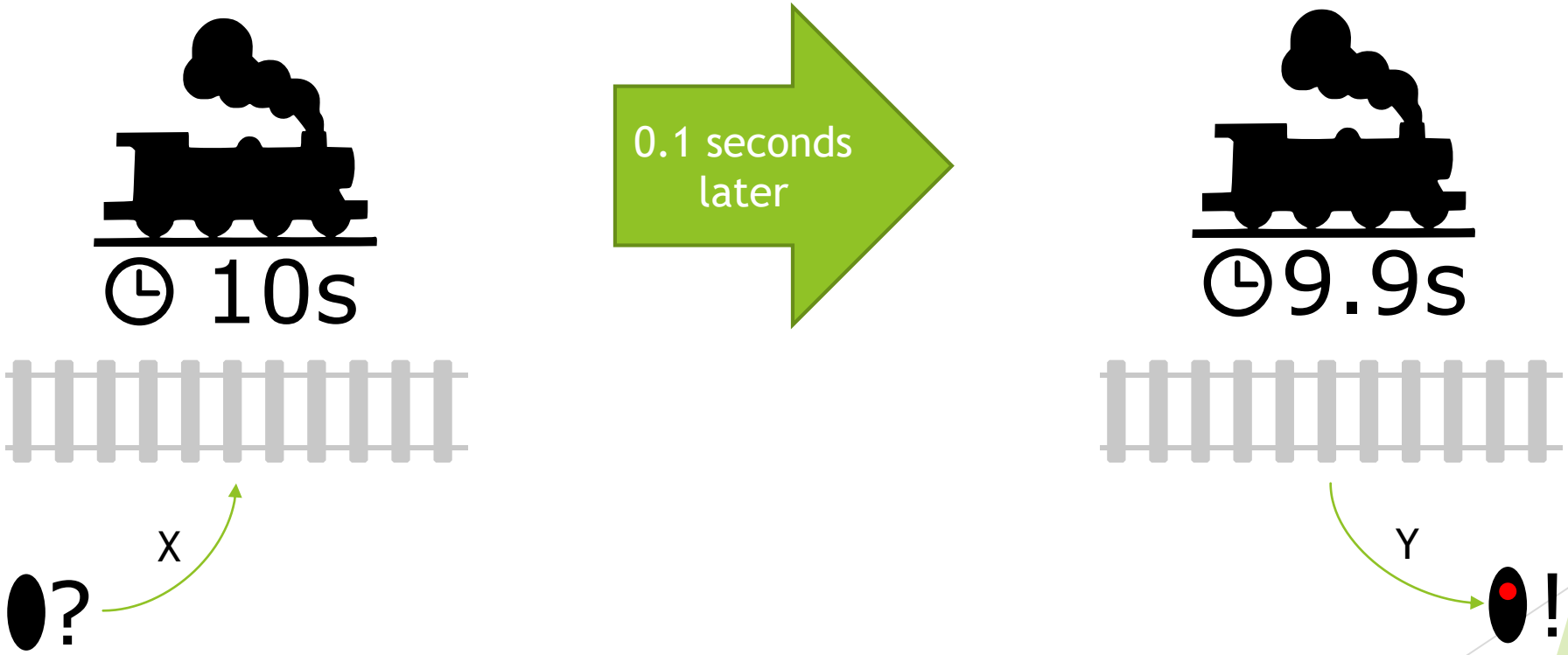
$$\lambda : S \to Y \cup \{\phi\}$$

$$ta : S \to \mathbb{R}^{+}_{0,+\infty}$$

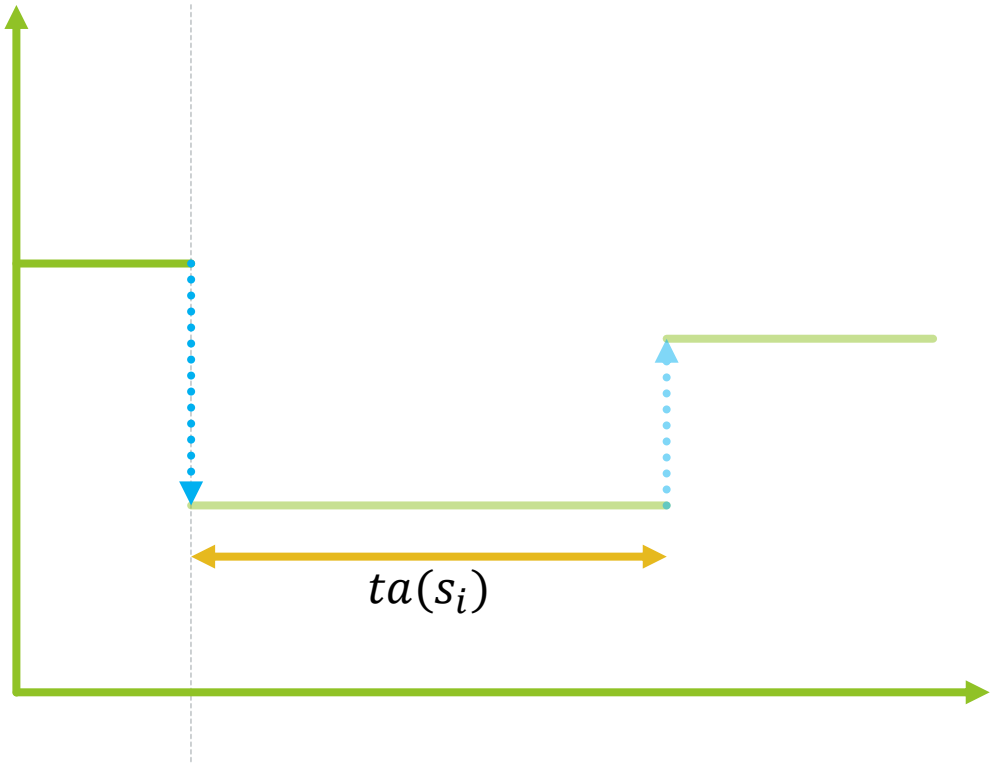
Pattern 1: Ignore an Event



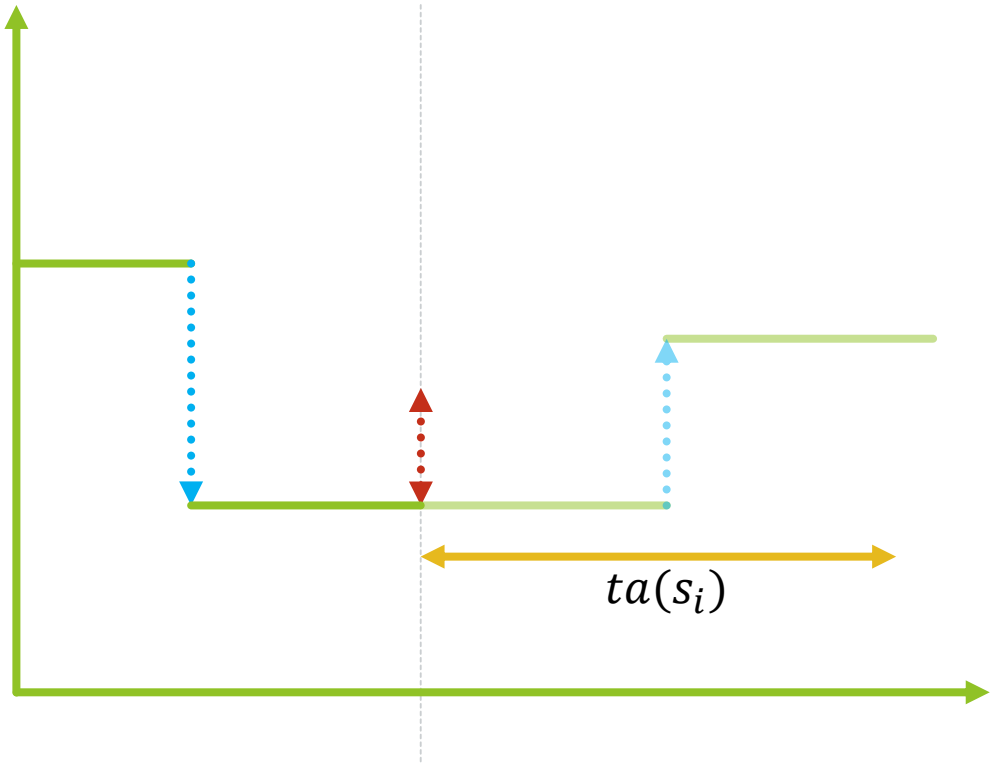
Pattern 1: Ignore an Event



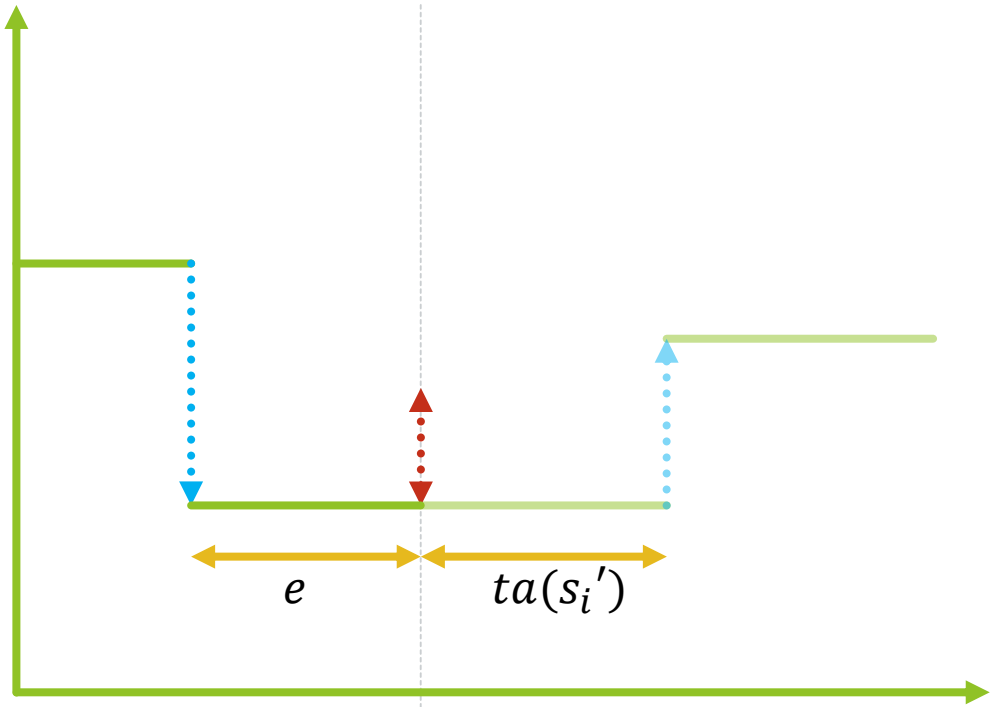
Pattern 1: Ignore an Event



Pattern 1: Ignore an Event



Pattern 1: Ignore an Event

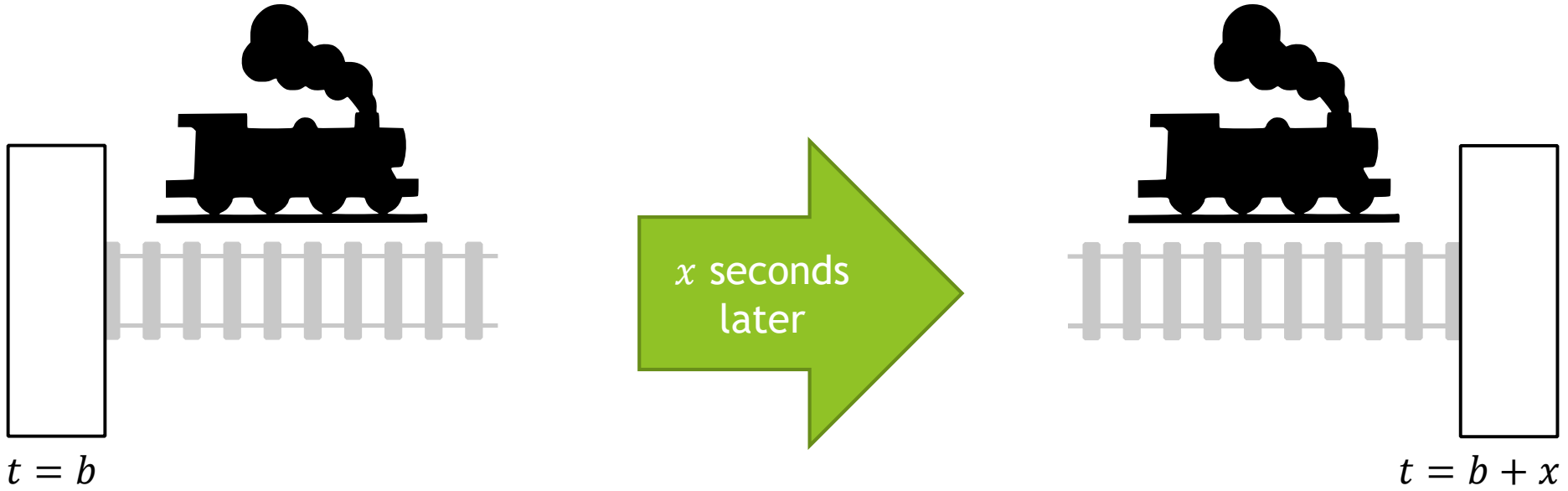


$$ta(s_i') = ta(s_i) - e$$

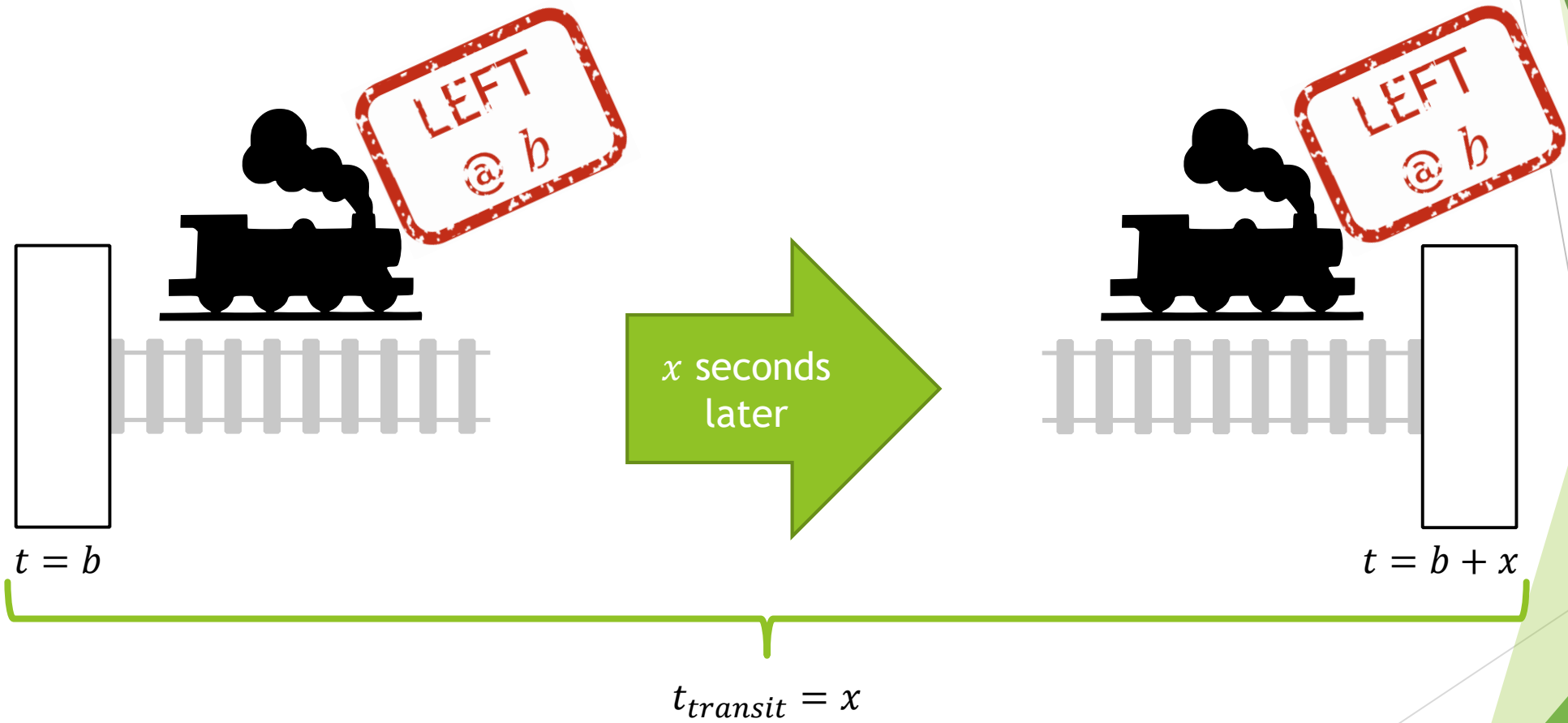
Pattern 1: Ignore an Event

```
class RailwaySegmentState():
    def __init__(self):
        self.timer = INFINITY

class RailwaySegment(AtomicDEVS):
    def extTransition(self, inputs):
        self.state.timer -= self.elapsed
        …

    def timeAdvance(self):
        return self.state.timer
```

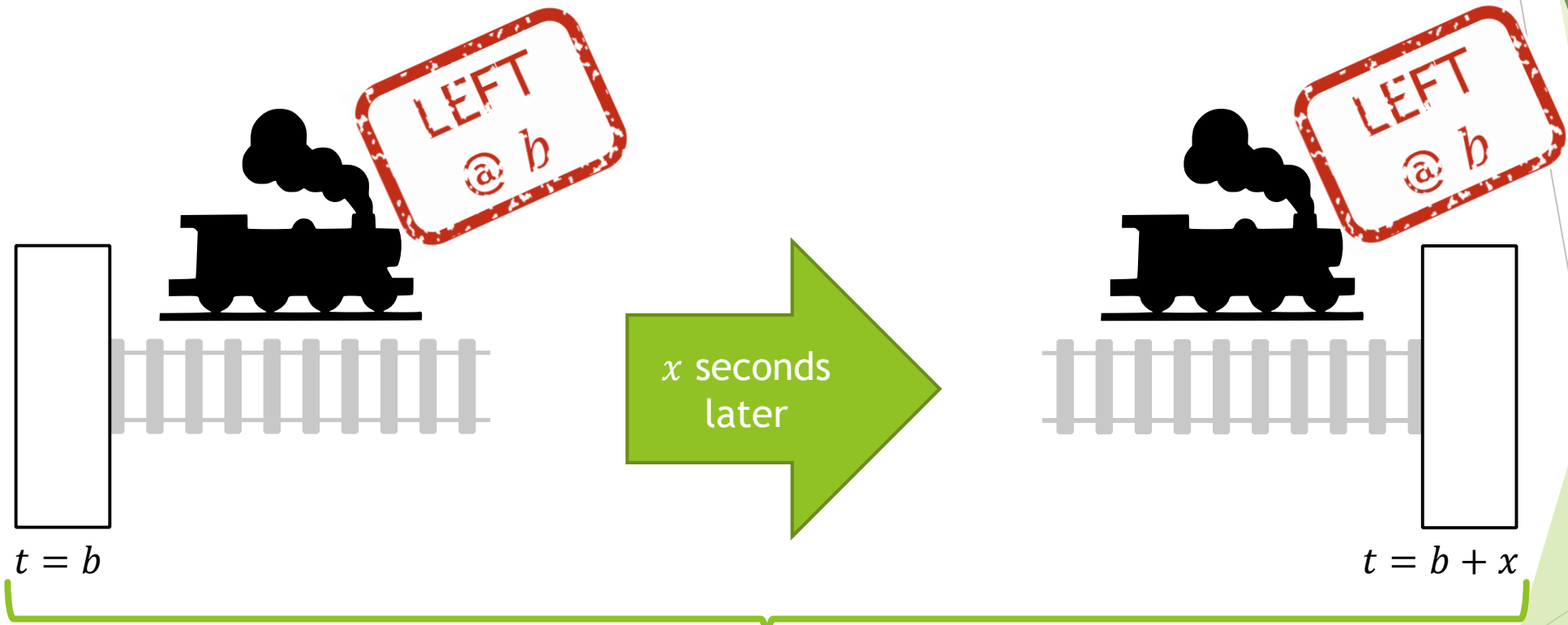
Pattern 2: Simulated Time



Pattern 2: Simulated Time

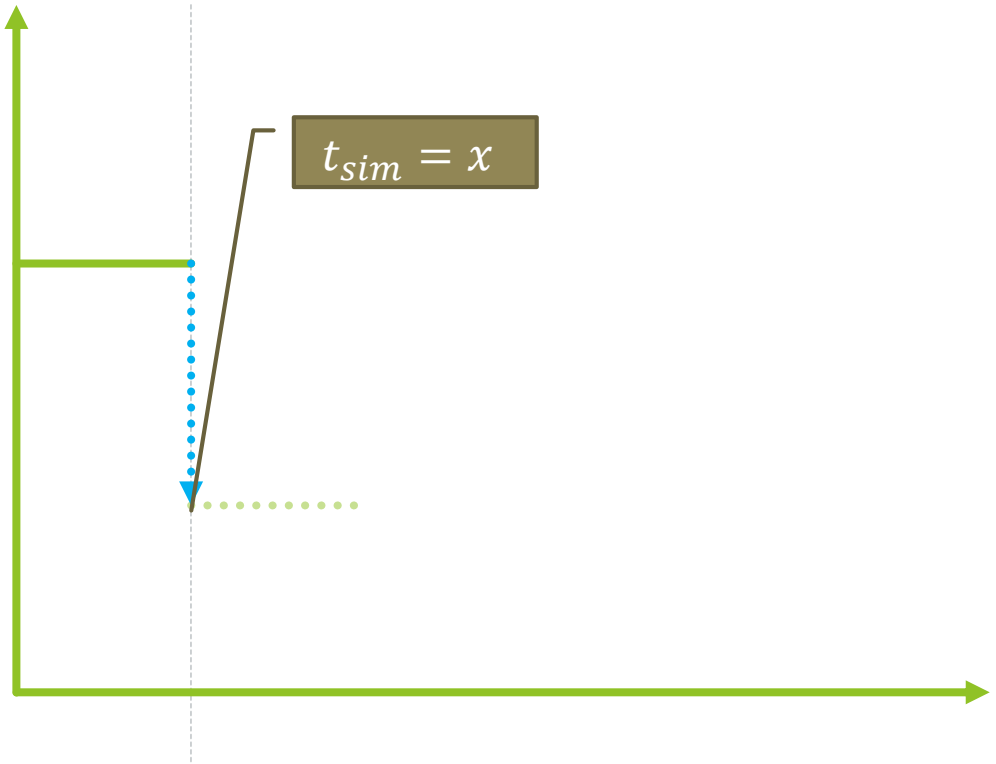


Pattern 2: Simulated Time

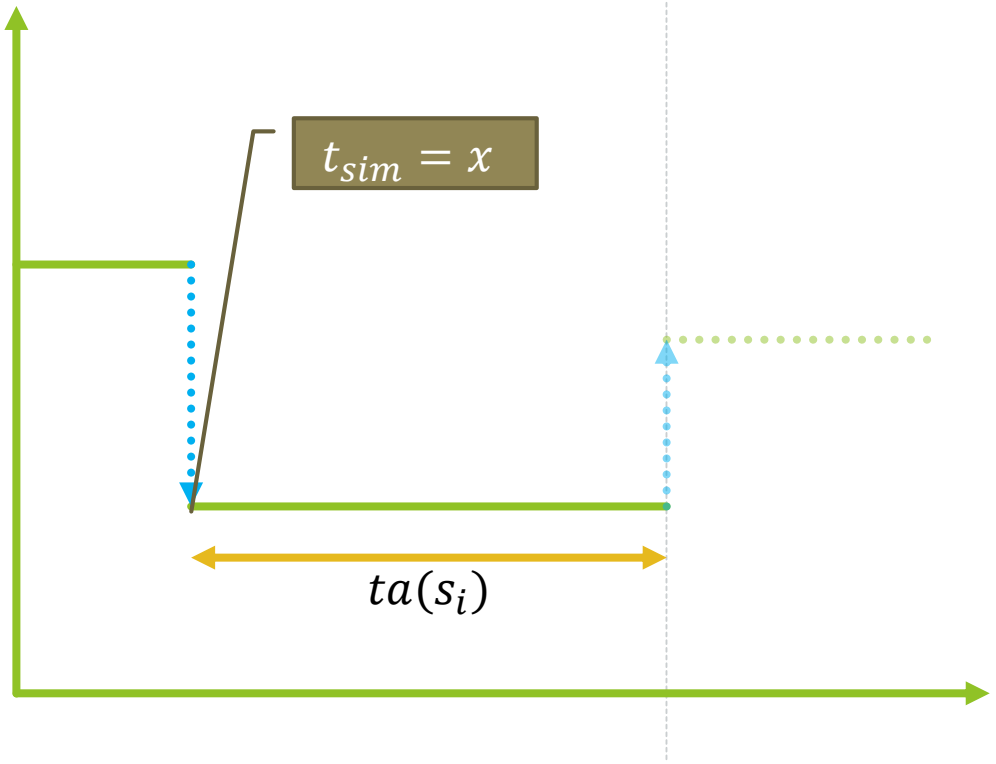


$$t = ?$$

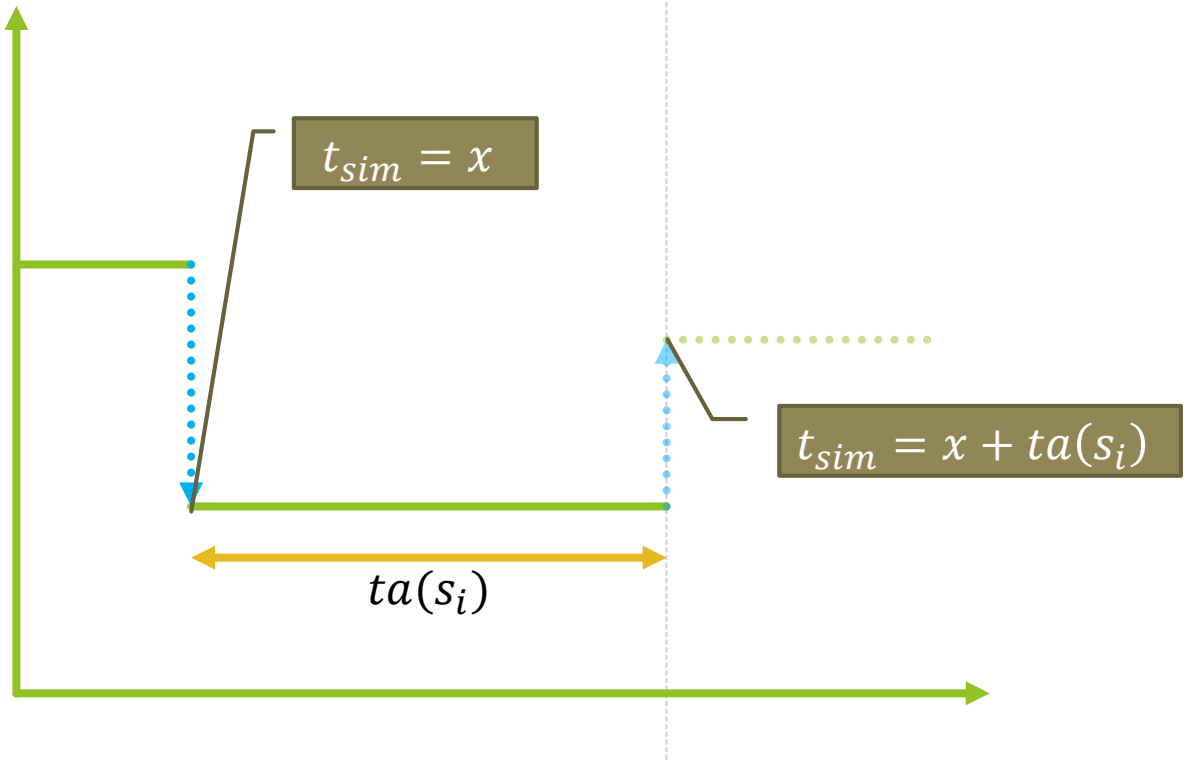
Pattern 2: Simulated Time



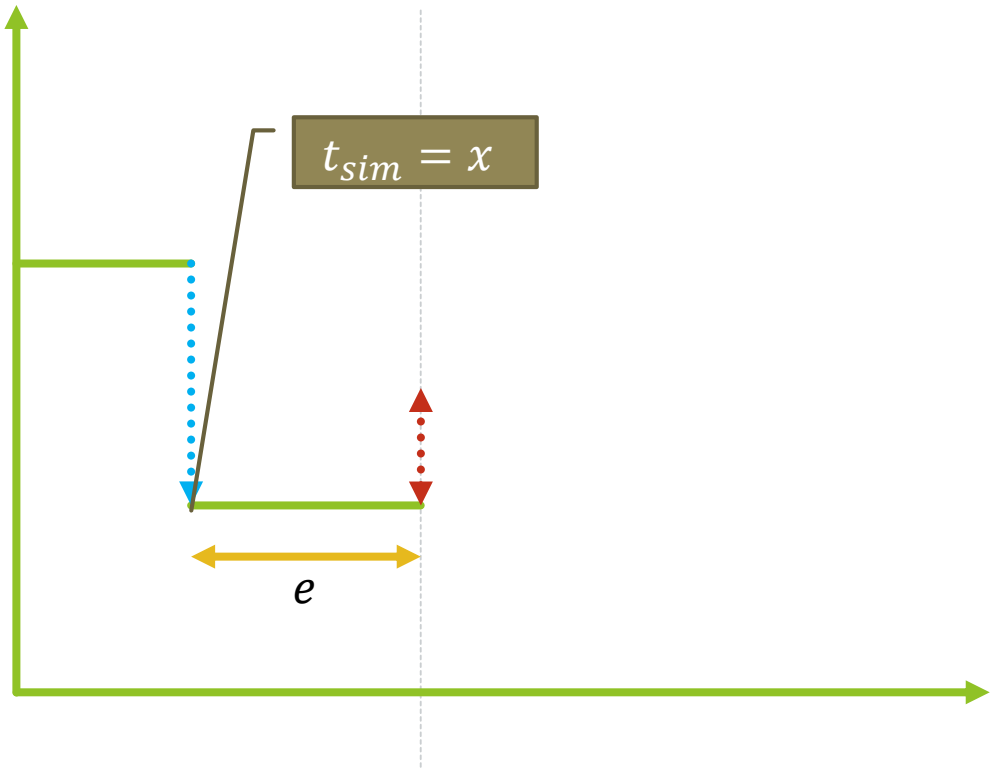
Pattern 2: Simulated Time



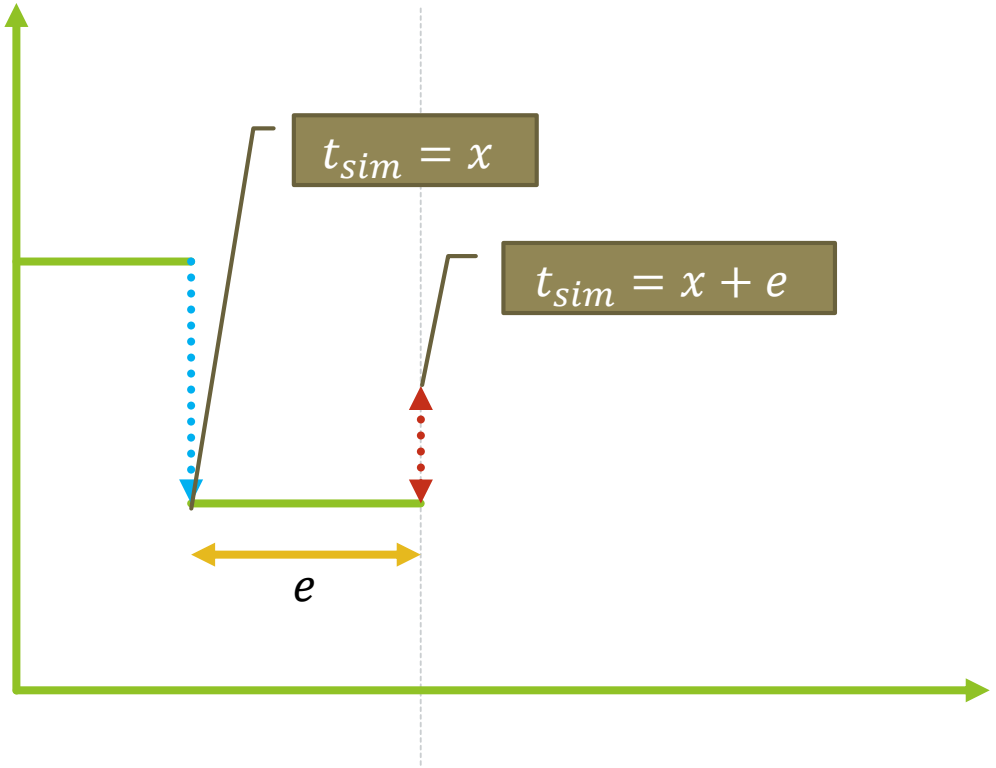
Pattern 2: Simulated Time



Pattern 2: Simulated Time



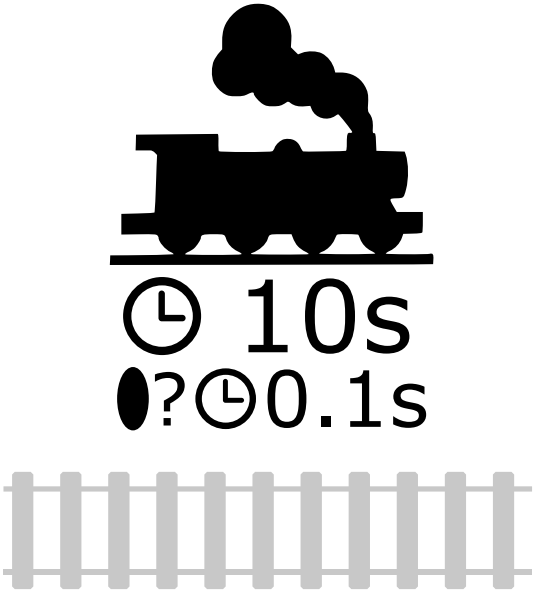
Pattern 2: Simulated Time



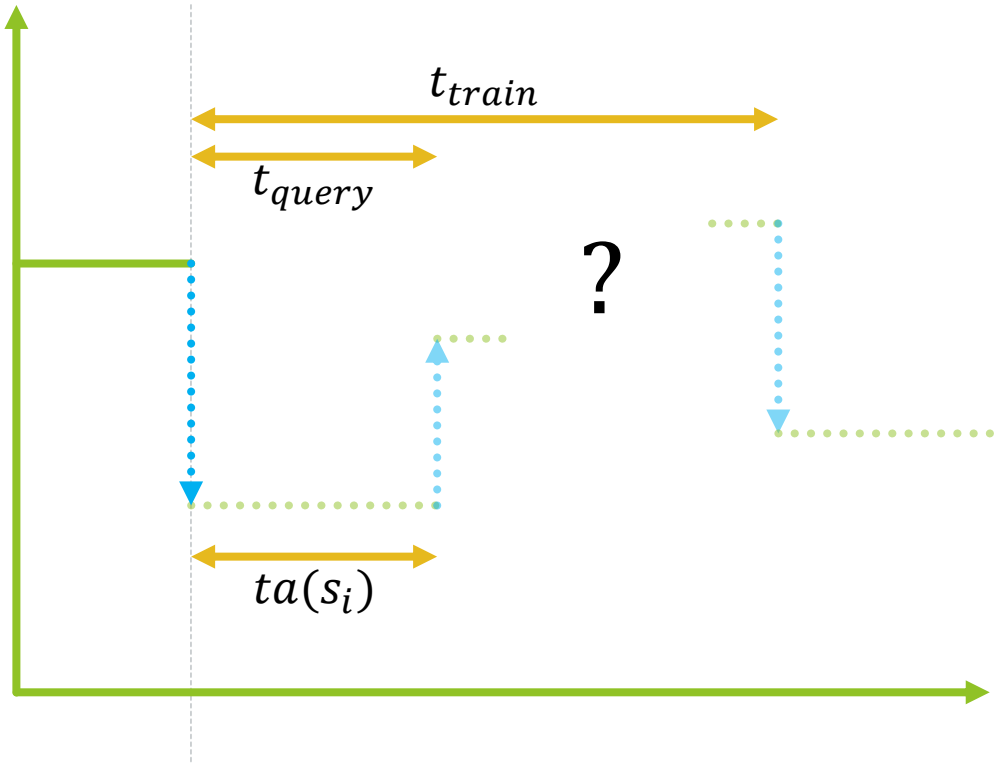
Pattern 2: Simulated Time

```
class GeneratorState:
    def __init__(self):
        self.t_sim = 0.0
        …

class Generator(AtomicDEVS):
    def __init__(self):
        self.state = GeneratorState()
        …

    def intTransition(self):
        self.state.t_sim += self.timeAdvance()
        …

    def extTransition(self, inputs):
        self.state.t_sim += self.elapsed
        …
```

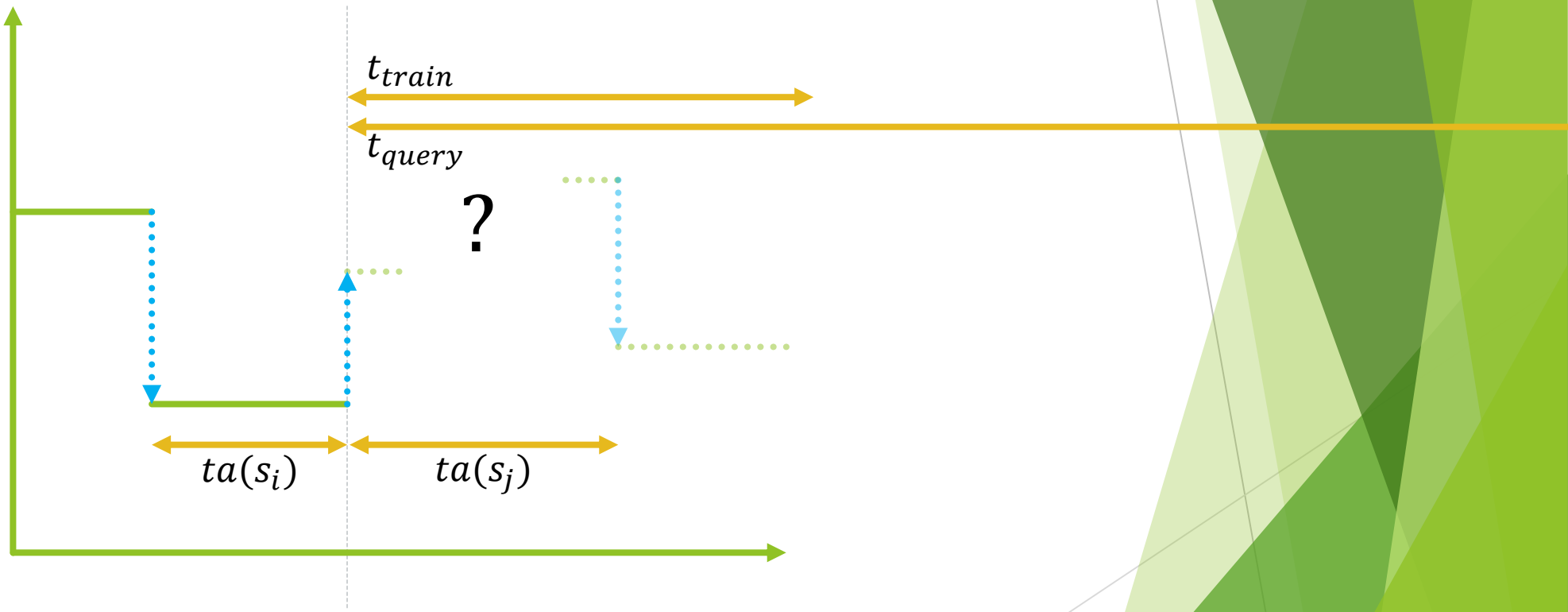
Pattern 3: Multiple Timers



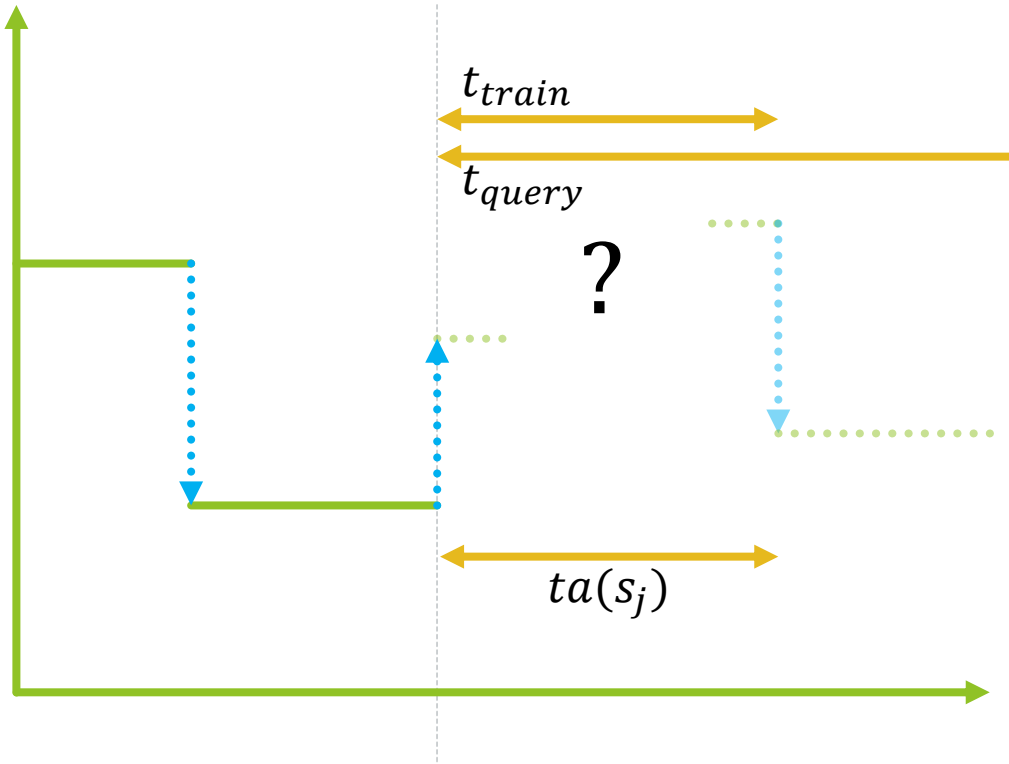
Pattern 3: Multiple Timers



Pattern 3: Multiple Timers



Pattern 3: Multiple Timers

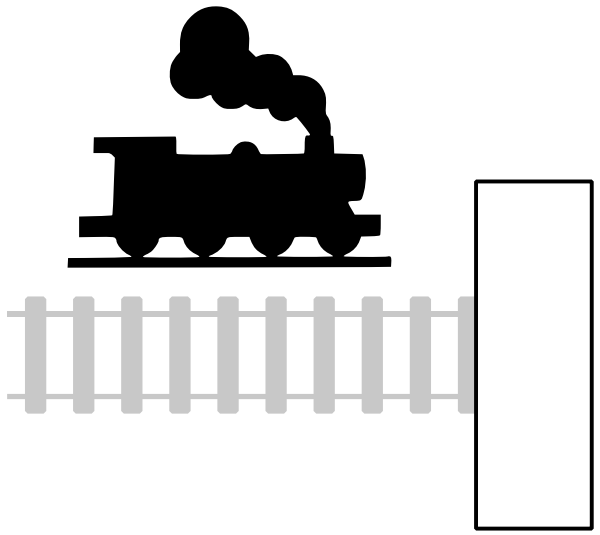


Pattern 3: Multiple Timers

```
class RailwaySegmentState:
    def __init__(self):
        self.t_query = INFINITY
        self.t_train = INFINITY

class RailwaySegment(AtomicDEVS):
    def timeAdvance(self):
        return min(self.state.t_query, \
                   self.state.t_train)

    def extTransition(self, inputs):
        self.state.t_query -= self.elapsed
        self.state.t_train -= self.elapsed
    …
```

```
    …

    def intTransition(self):
        self.state.t_query -= self.timeAdvance()
        self.state.t_train -= self.timeAdvance()
        if (self.state.t_query == 0):
            … # process query
        elif (self.state.t_train == 0):
            … # process train
```

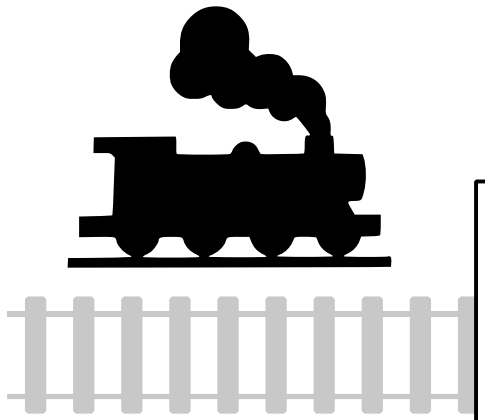
Pattern 4: Statistics Gathering



n trains = n doubles = $8n$ bytes

[1815.7; 1901.1; 1801.4; 1867.3; 1911.8; 1846.4; 1844.3; 1873.5; …]

Pattern 4: Statistics Gathering



n trains = n doubles = $8n$ bytes

[1815.7; 1901.1; 1801.4; 1867.3; 1.8; 1846.4; 1844.3; 1873.5; …]

$$sum = \sum_i t_{travel} = 36{,}902{,}140.45$$

$$count = |arrivals| = 20{,}000$$

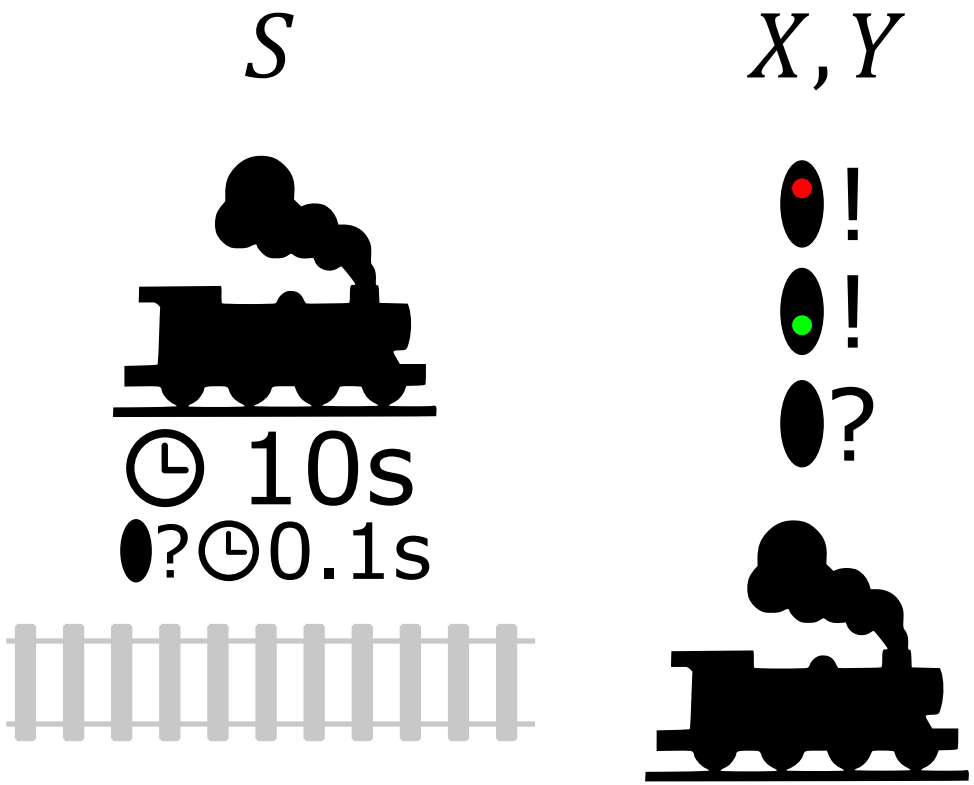
$$avg = \frac{sum}{count}$$

n trains = 1 double + 1 long long int = 16 bytes

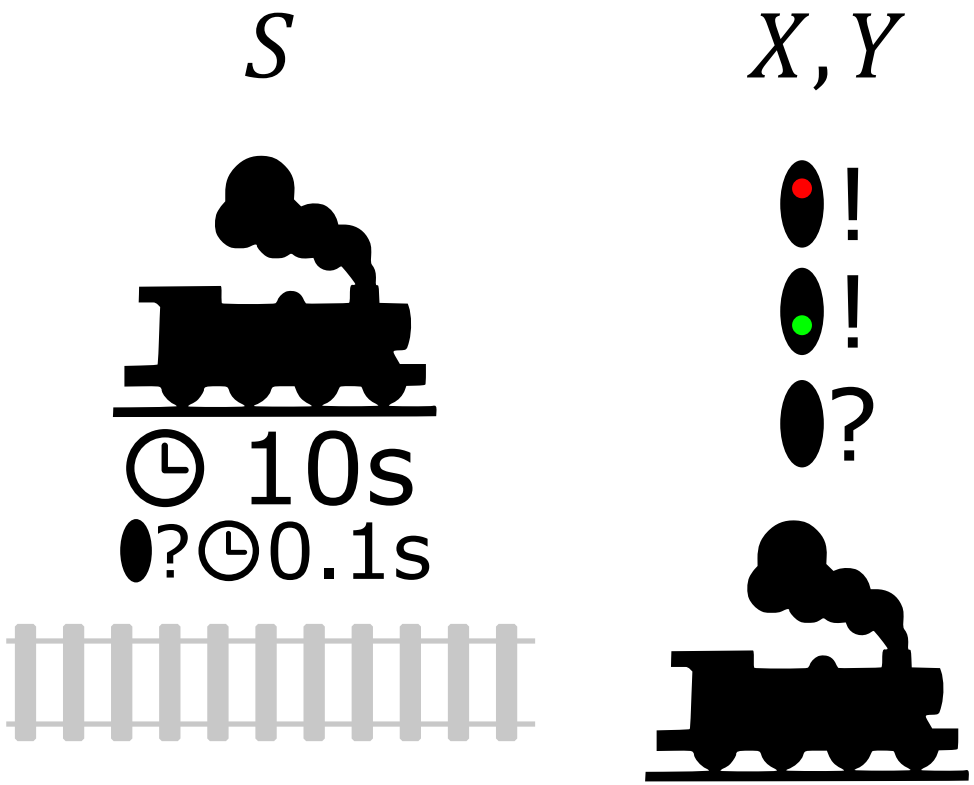
Pattern 4: Statistics Gathering

```
class CollectorState():
    def __init__(self):
        self.counter = 0
        self.accum = 0.0
        self.t_sim = 0.0

class Collector(AtomicDEVS):
    def extTransition(self, inputs):
        left = inputs[self.train].left
        transit = self.state.t_sim - left
        self.accum += transit
        self.counter += 1
```

Pattern 5: Complex State/Event



Pattern 5: Complex State/Event



$$S = \{\langle timer_{train}, timer_{query}, v_{train}, t_{train}, a_{train}\rangle | timer_{train} \in \mathbb{R}^+, \dots\}$$

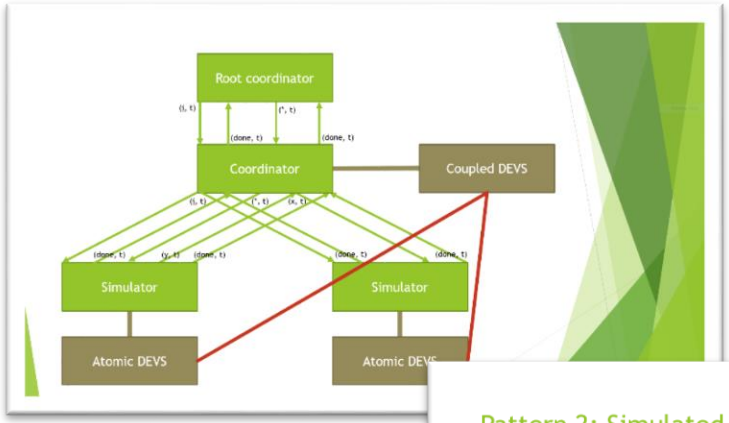
$$X, Y = \{\langle t, v, a\rangle | t \in \mathbb{R}^+, \dots\} \cup \{\text{query}\} \cup \{colour | colour \in \{red, green\}\}$$

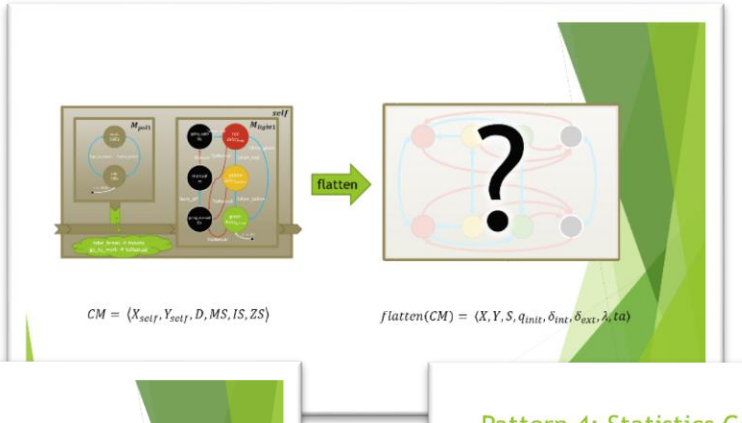
Pattern 5: Complex State/Event

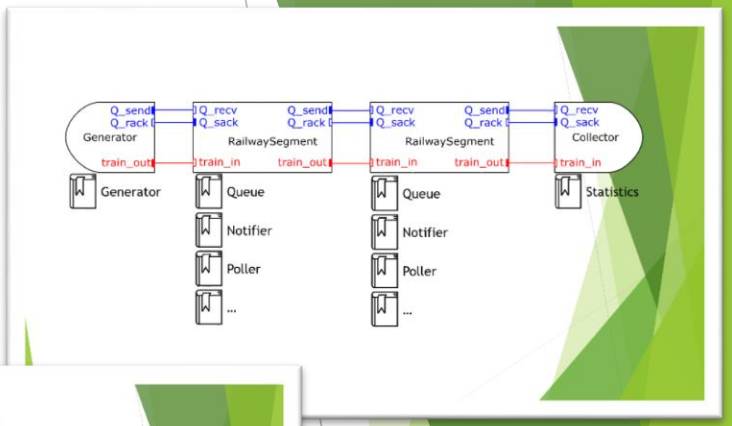
```
class RailwaySegmentState:
    def __init__(self):
        self.train = None

class RailwaySegment(AtomicDEVS):
    def __init__(self):
        self.state = RailwaySegmentState()
        …

    def extTransition(self, inputs):
        …
        self.state.train = inputs[self.train_in]
        …
```

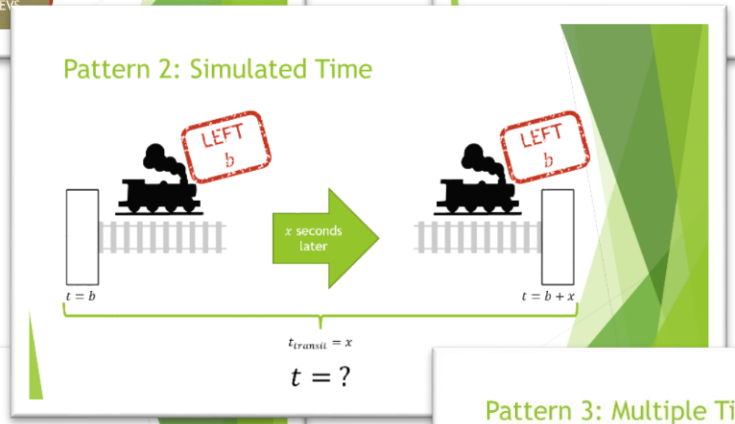
```
class Train:
    def __init__(self, t, a):
        self.t = t
        self.a = a
        self.v = 0.0

class Query:
    def __init__(self):
        pass

class QueryAck:
    def __init__(self, colour):
        self.colour = colour
```

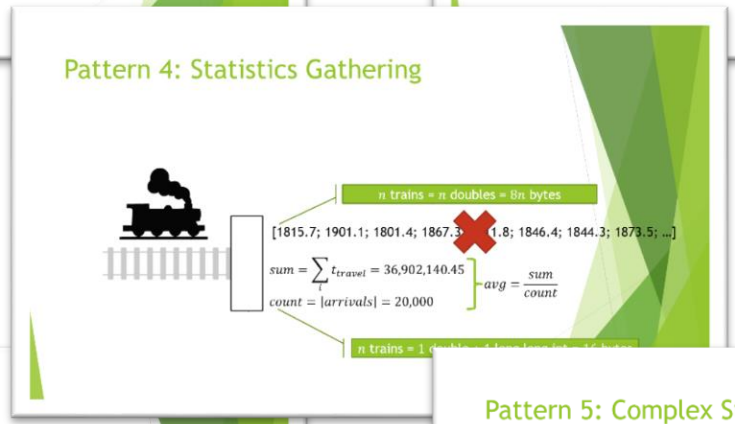
Conclusion

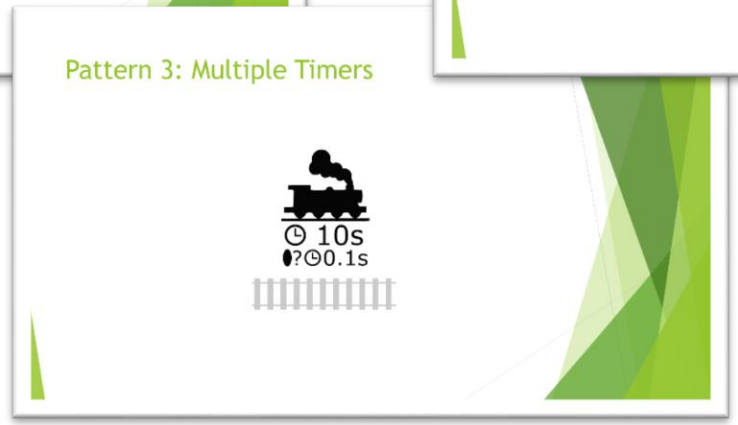


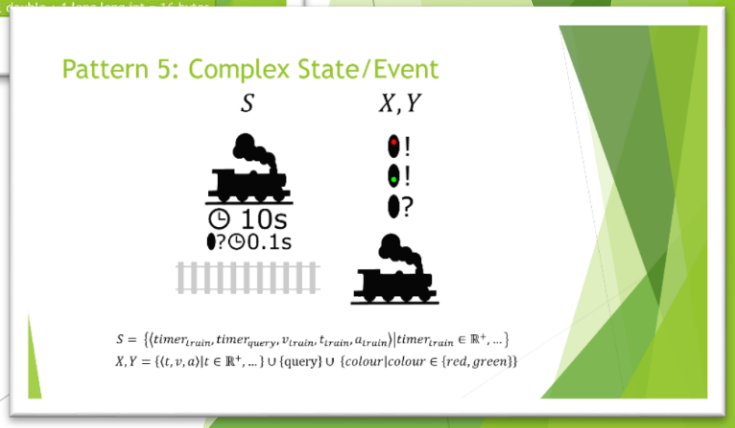












http://msdl.cs.mcgill.ca/projects/PythonPDEVS