Presenting *RAPS: R Aid for P Systems*

+

Simulating stochastic algorithms

•••

19th Brainstorming Week on Membrane Computing

José Antonio Rodríguez Gallego

Presenting *RAPS: R Aid for P Systems*

•••

19th Brainstorming Week on Membrane Computing

José Antonio Rodríguez Gallego

The *RAPS* package

R Aid for P Systems

The *RAPS* package

Dependencies & Installation

The *RAPS* package

Dependencies & Installation

The *RAPS* package

Dependencies & Installation

The *RAPS* package

Dependencies & Installation


```
devtools::github_install()
```

The *RAPS* package

Dependencies & Installation

CLICK HERE!

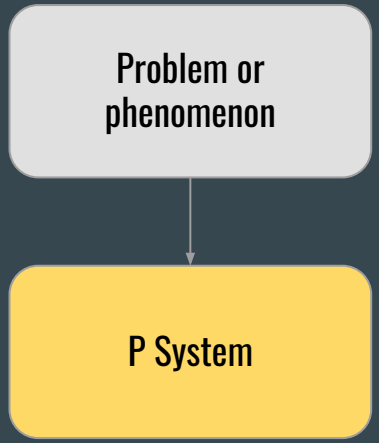
The *RAPS* package

Use + Objectives

Problem or phenomenon

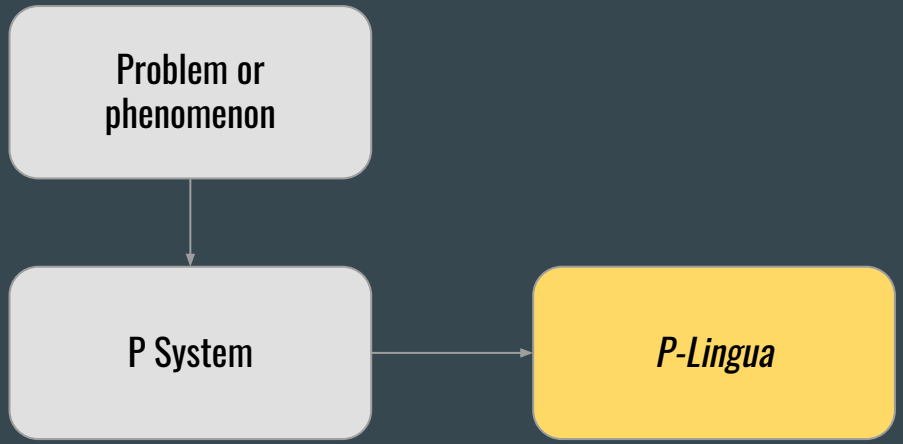
The *RAPS* package

Use + Objectives



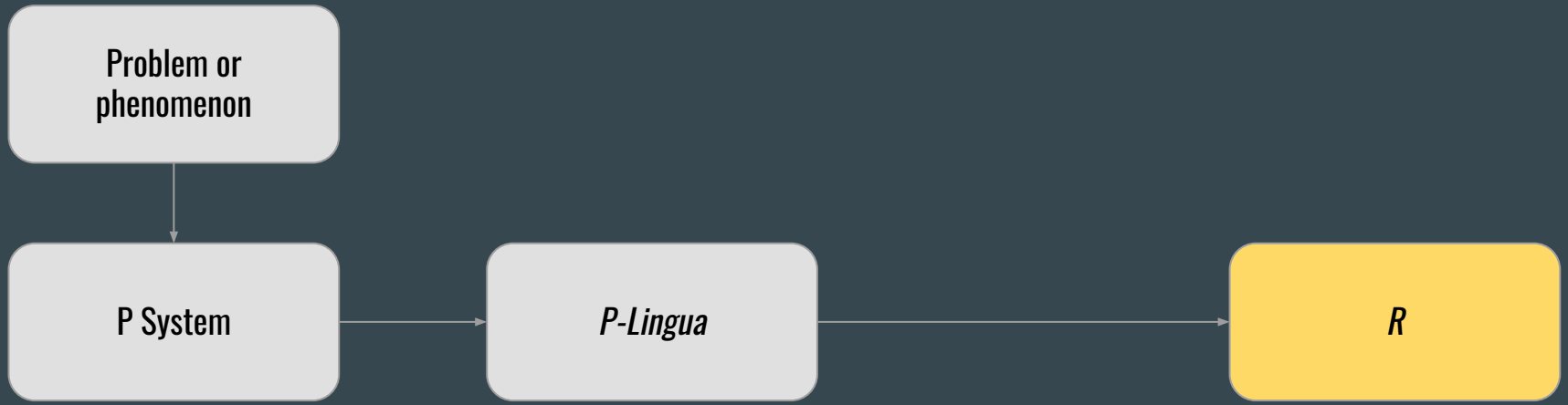
The *RAPS* package

Use + Objectives



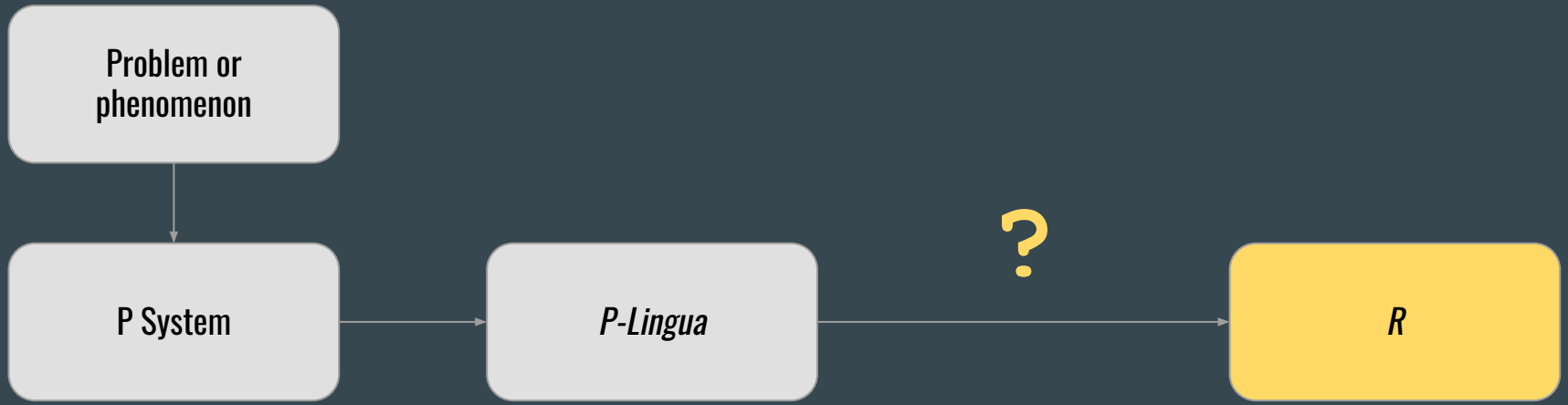
The *RAPS* package

Use + Objectives



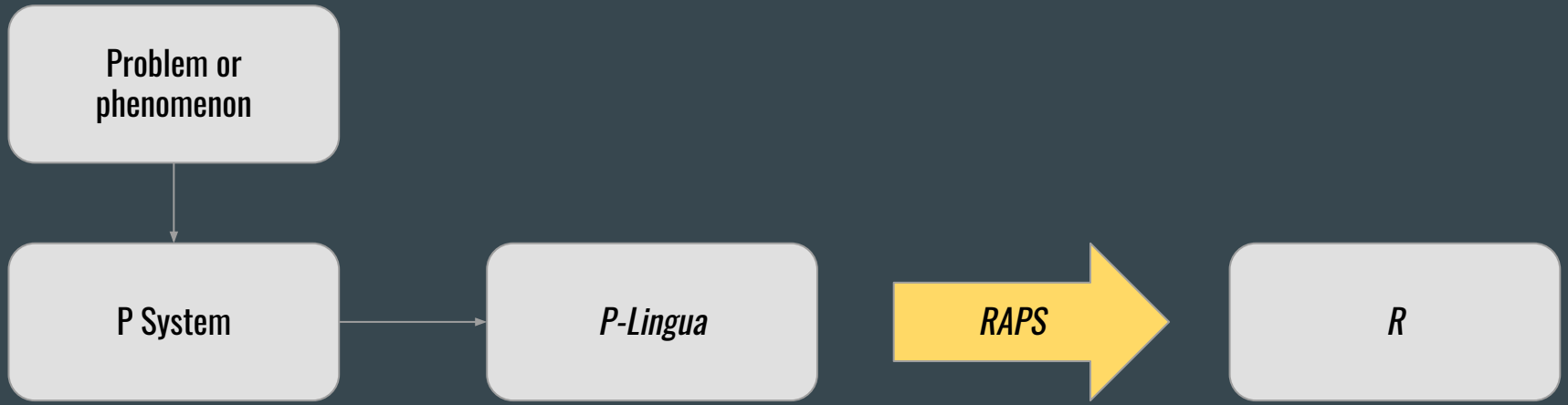
The *RAPS* package

Use + Objectives



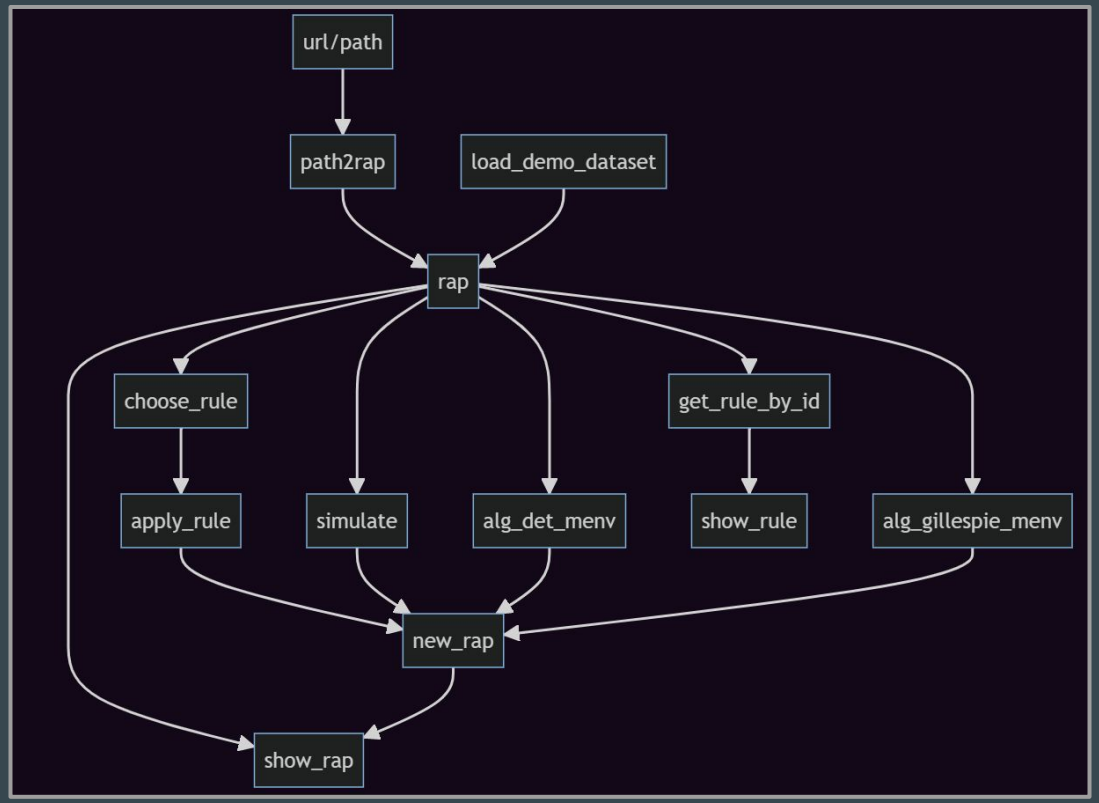
The *RAPS* package

Use + Objectives



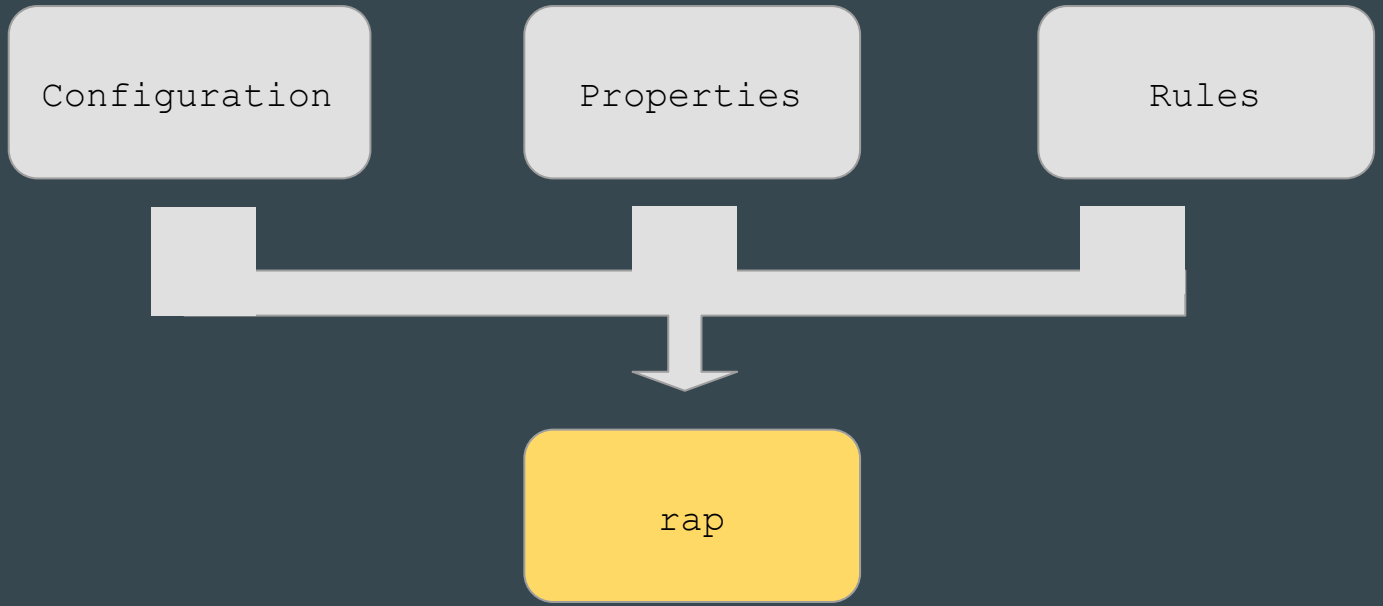
The *RAPS* package

Use + Objectives



The *RAPS* package

The `rap` (*Representing A P system*) object



The *RAPS* package

rap - Configuration

Environment	ID	Label	Objects	SuperM	SubM	Charge	Other_params
1	1	1	[(a, 1)]	0	[2,3]	-1	@immutable
1	2	2	[(b, 2), (c,3)]	1	NULL	+1	NA
1	3	3	[(@filler, 1)]	1	NULL	+1	NA

The *RAPS* package

rap - Properties

- objects_dictionary, labels_dictionary, strings_dictionary
- features (“pattern”, “sc”)
- max_multiplicity, n_rules, output_version, model_id

The *RAPS* package

Rules & Syntax

$$[a\ [b]_2\ [\ ]_3 \rightarrow a'\ [b']_2\ ]_1$$

The *RAPS* package

Rules & Syntax

$$[a\ [b]_2\ [\ ]_3 \rightarrow a'\ [b']_2\ ]_1$$

rule_id	dissolves	priority	main_membrane_label	lhs	rhs	propensity
value0	FALSE	-	1	lhs_1	rhs_1	1974

	where	object	multiplicity
lhs =	@here	a	1
	mem_2	b	2
	@exists	mem_3	1
rhs =	@here	ap	2
	mem_2	bp	3

The *RAPS* package

Rules & Syntax

$$[a\ [b]_2\ [\ ]_3 \rightarrow a'\ [b']_2\ ]_1$$

rule_id	dissolves	priority	main_membrane_label	lhs	rhs	propensity
value0	FALSE	-	1	lhs_1	rhs_1	1974

	where	object	multiplicity
lhs =	@here	a	1
	mem_2	b	2
	@exists	mem_3	1
rhs =	@here	ap	2
	mem_2	bp	3

The *RAPS* package

Rules & Syntax

$$[a\ [b]_2\ [\ ]_3 \rightarrow a'\ [b']_2\ ]_1$$

rule_id	dissolves	priority	main_membrane_label	lhs	rhs	propensity
value0	FALSE	-	1	lhs_1	rhs_1	1974

	where	object	multiplicity
lhs =	@here	a	1
	mem_2	b	2
	@exists	mem_3	1
rhs =	@here	ap	2
	mem_2	bp	3

The *RAPS* package

Rules & Syntax

$$[a\ [b]_2\ [\ ]_3 \rightarrow a'\ [b']_2\ ]_1$$

rule_id	dissolves	priority	main_membrane_label	lhs	rhs	propensity
value0	FALSE	-	1	lhs_1	rhs_1	1974

	where	object	multiplicity
lhs =	@here	a	1
	mem_2	b	2
	@exists	mem_3	1
rhs =	@here	ap	2
	mem_2	bp	3

The *RAPS* package

Rules & Syntax

$$[a\ [b]_2\ [\ ]_3 \rightarrow a'\ [b']_2\ ]_1$$

rule_id	dissolves	priority	main_membrane_label	lhs	rhs	propensity
value0	FALSE	-	1	lhs_1	rhs_1	1974

	where	object	multiplicity
lhs =	@here	a	1
	mem_2	b	2
	@exists	mem_3	1
rhs =	@here	ap	2
	mem_2	bp	3

The *RAPS* package

Reading functions

`path2rap()`

`load_demo_dataset()`

The *RAPS* package

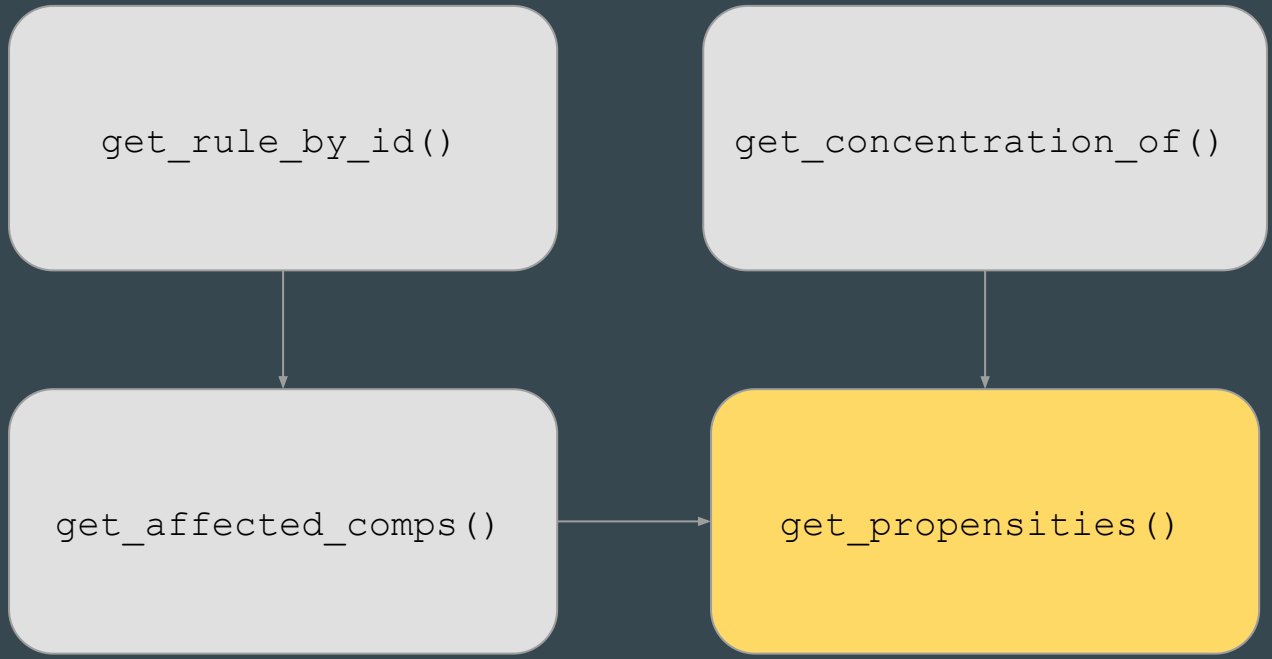
Simulation - Applying the rules

```
is_applicable()
check_applicability()
```

```
apply_rule()
apply_rule_menv()
```

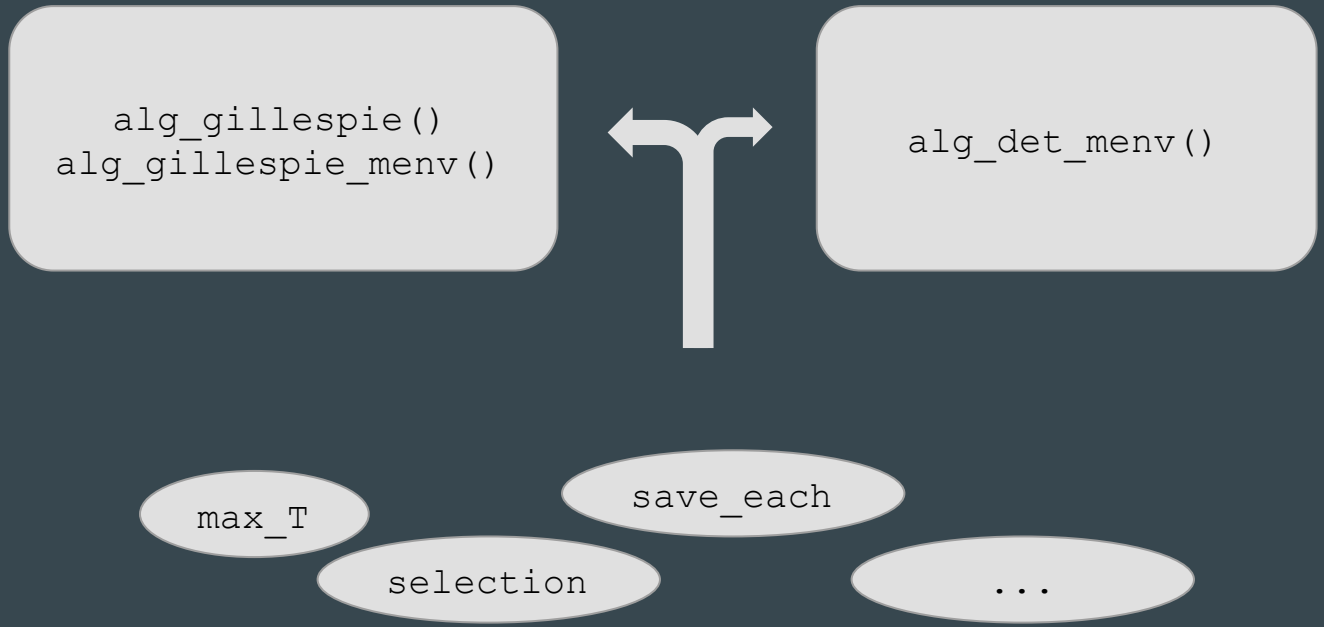
The *RAPS* package

Simulation - Propensities



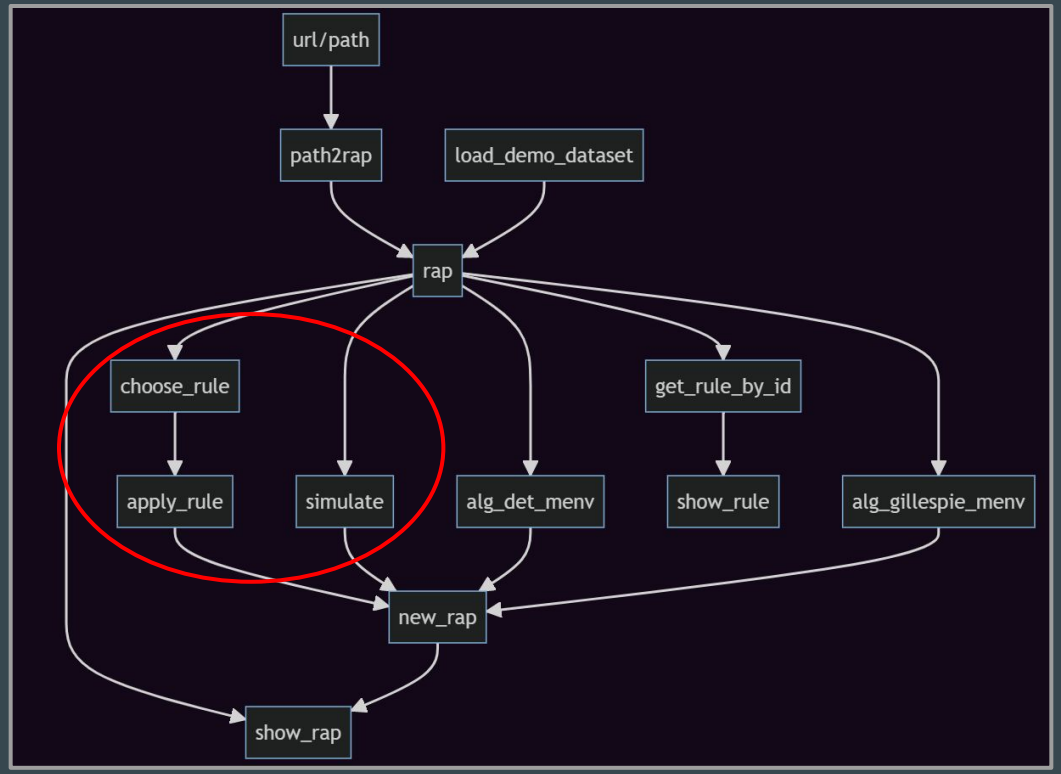
The *RAPS* package

Simulation - Classical algorithms



The *RAPS* package

Custom simulators



The *RAPS* package

Visualization

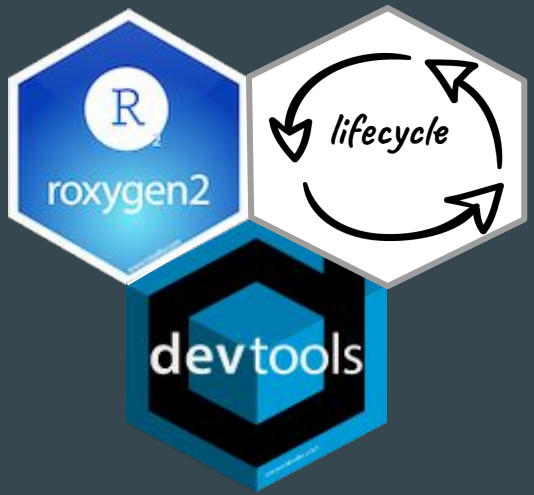
`show_rule()`

`show_rap()`

`get_concentration_of()`

The *RAPS* package

Some other things

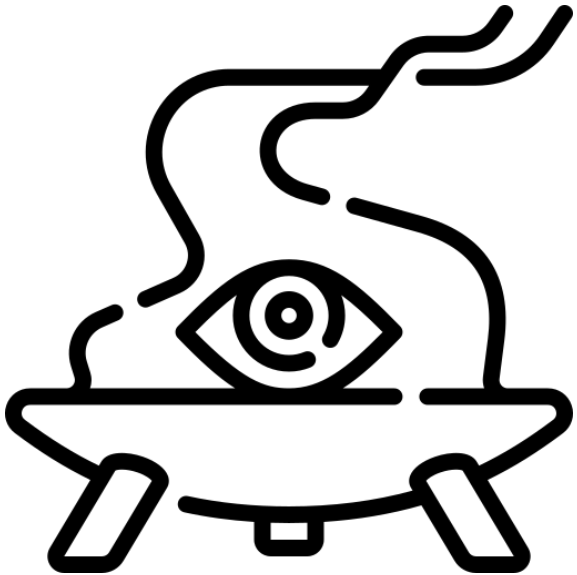


debug

verbose

demos/

Future work



Future work

- **Improvement of the simulators**
- Validation of the simulators
- `path2rap()`
- *CRAN*
- `rap` visualization
- Other rules
- Virus machines (???)

Simulating stochastic algorithms

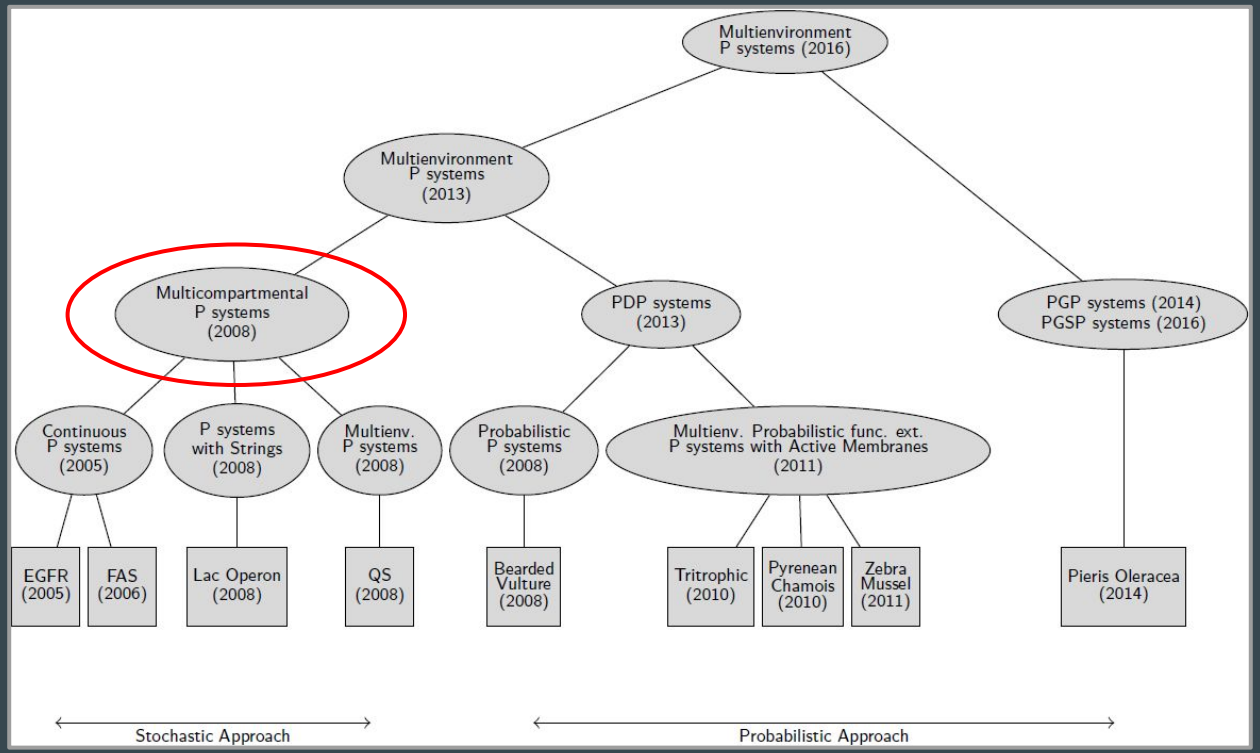
19th Brainstorming Week on Membrane Computing

José Antonio Rodríguez Gallego

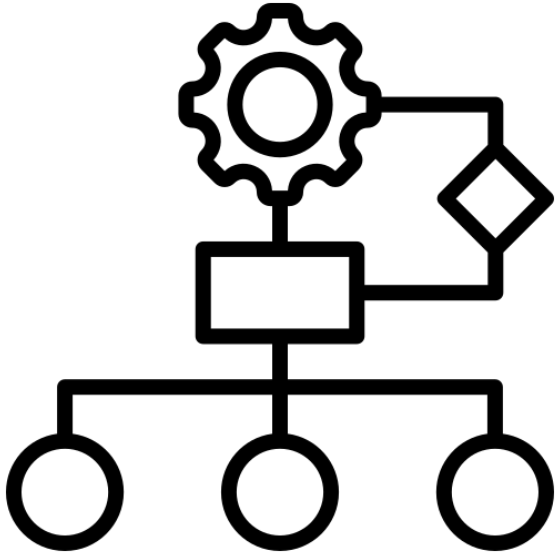
Where are we?

Where are we?

What do we want to simulate?

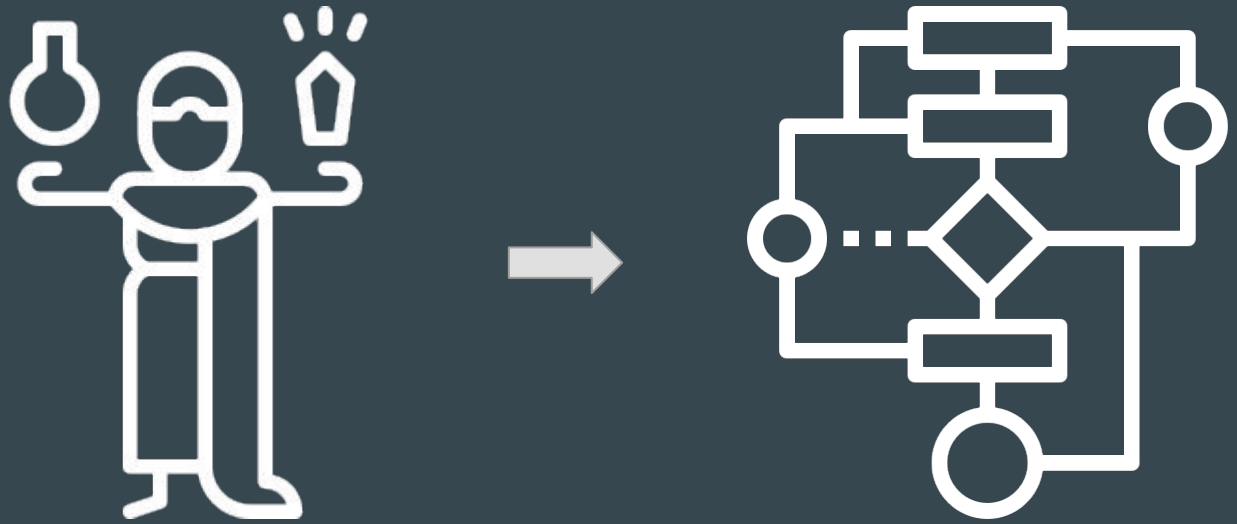


Classical Simulation Algorithms



Classical Simulation Algorithms

Propensities



Classical Simulation Algorithms

Propensities

Definition (Propensity): We define a propensity $p_j(X_i)dt$ as the probability given to the execution of a rule r_j in the interval $[t, t+dt)$, beginning the system at state X_i.

Classical Simulation Algorithms

Propensities

Definition (Propensity): We define a propensity $p_j(X_i)dt$ as the probability given to the execution of a rule r_j in the interval $[t, t+dt)$, beginning the system at state X_i.

Kinetic constant

k_j

Classical Simulation Algorithms

Propensities

Definition (Propensity): We define a propensity $p_j(X_i)dt$ as the probability given to the execution of a rule r_j in the interval $[t, t+dt)$, beginning the system at state X_i.

Kinetic constant → Stochastic constant

k_j

$$c_j = \begin{cases} k_j & \text{if } r_j \equiv (s_i \to p) \\ \begin{cases} \frac{k_j}{V} & \text{if } i_1 \neq i_2 \\ \frac{2 \cdot k_j}{V} & \text{if } i_1 \neq i_2 \end{cases} & \text{if } r_j \equiv (s_{i_1} + s_{i_1} \to p) \end{cases}$$

Classical Simulation Algorithms

Propensities

Definition (Propensity): We define a propensity $p_j(X_i)dt$ as the probability given to the execution of a rule r_j in the interval $[t, t+dt)$, beginning the system at state X_i.

Kinetic constant → Stochastic constant → Propensity

$$p_j(X(t)) = \begin{cases} c_j \cdot X_i(t) & \text{if } r_j \equiv (s_i \to p) \\ \begin{cases} c_j \cdot X_i(t) \cdot X_{i'}(t) & \text{if } i \neq i' \\ c_j \cdot \frac{1}{2} \cdot X_i(t) \cdot (x_{i'}(t) - 1) & \text{if } i = i' \end{cases} & \text{if } r_j \equiv (s_i + s_{i'} \to p) \end{cases}$$

$$p_j = k_j \cdot \prod_{i \in lhs} X_i$$

Classical Simulation Algorithms

Propensities

Definition (Propensity): We define a propensity $p_j(X_i)dt$ as the probability given to the execution of a rule r_j in the interval $[t, t+dt)$, beginning the system at state X_i.

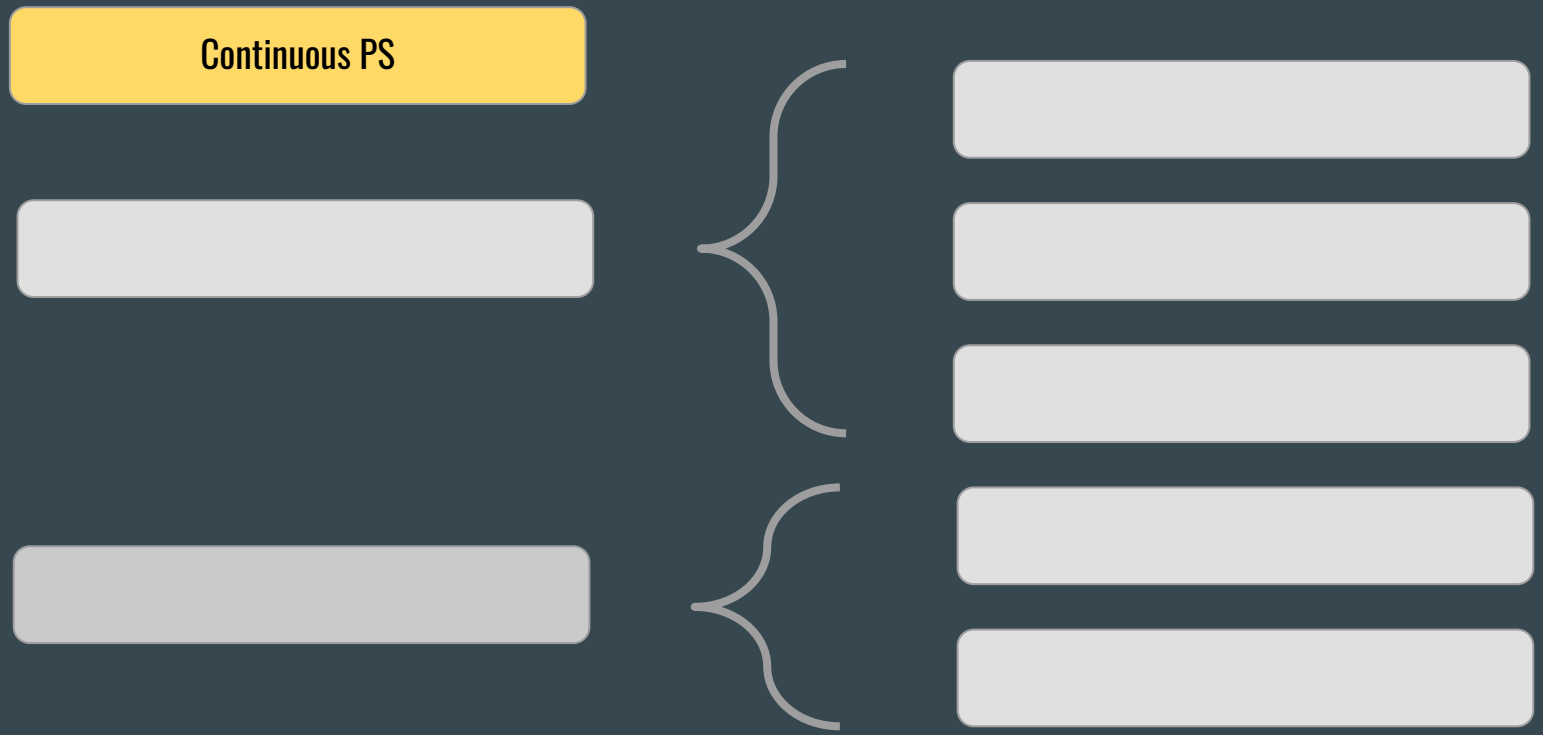
Kinetic constant → Stochastic constant → Propensity

$$p_j(X(t)) = \begin{cases} c_j \cdot X_i(t) & \text{if } r_j \equiv (s_i \to p) \\ \begin{cases} c_j \cdot X_i(t) \cdot X_{i'}(t) & \text{if } i \neq i' \\ c_j \cdot \frac{1}{2} \cdot X_i(t) \cdot (x_{i'}(t) - 1) & \text{if } i = i' \end{cases} & \text{if } r_j \equiv (s_i + s_{i'} \to p) \end{cases}$$

$$p_j = k_j \cdot \prod_{i \in lhs} X_i$$

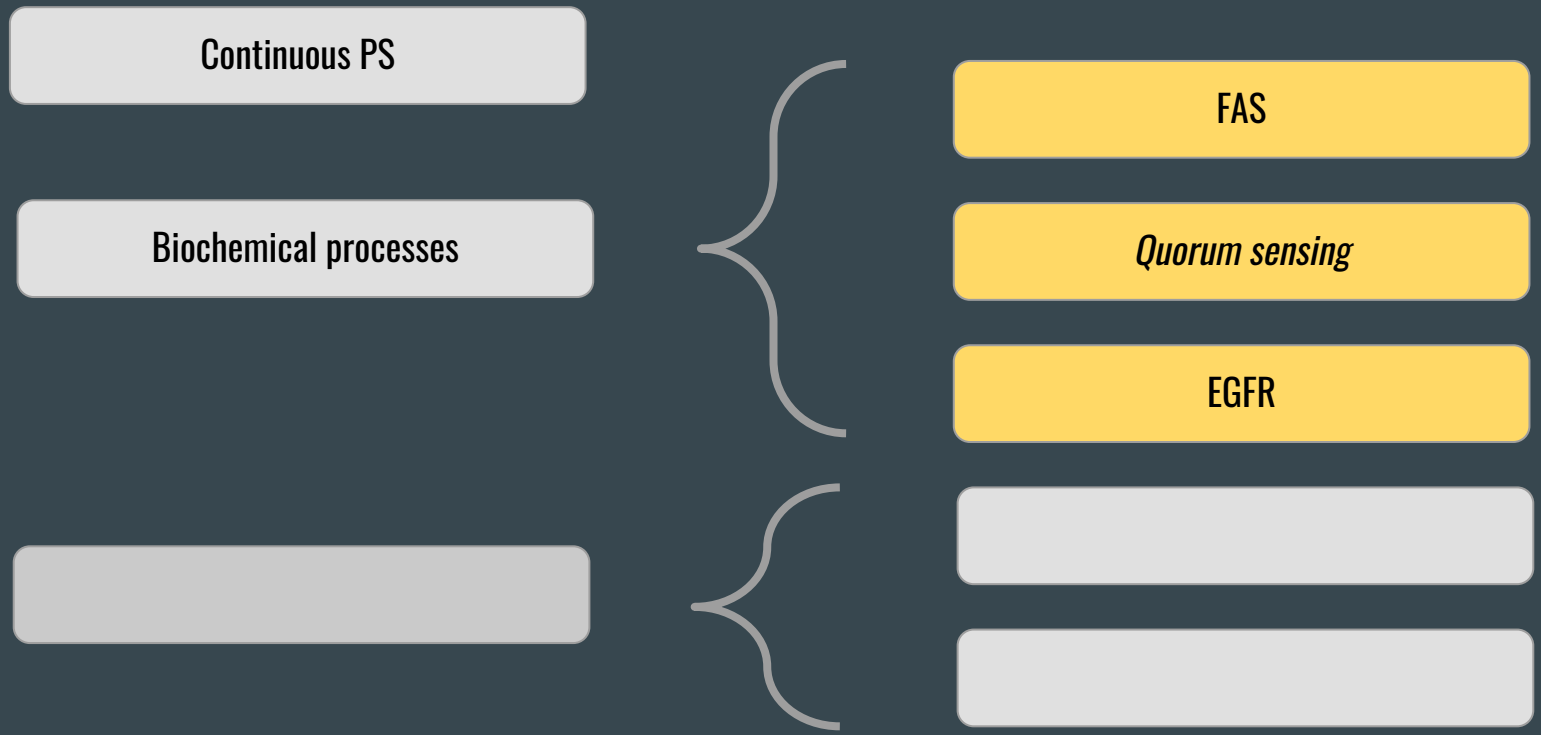
Classical Simulation Algorithms

Study cases



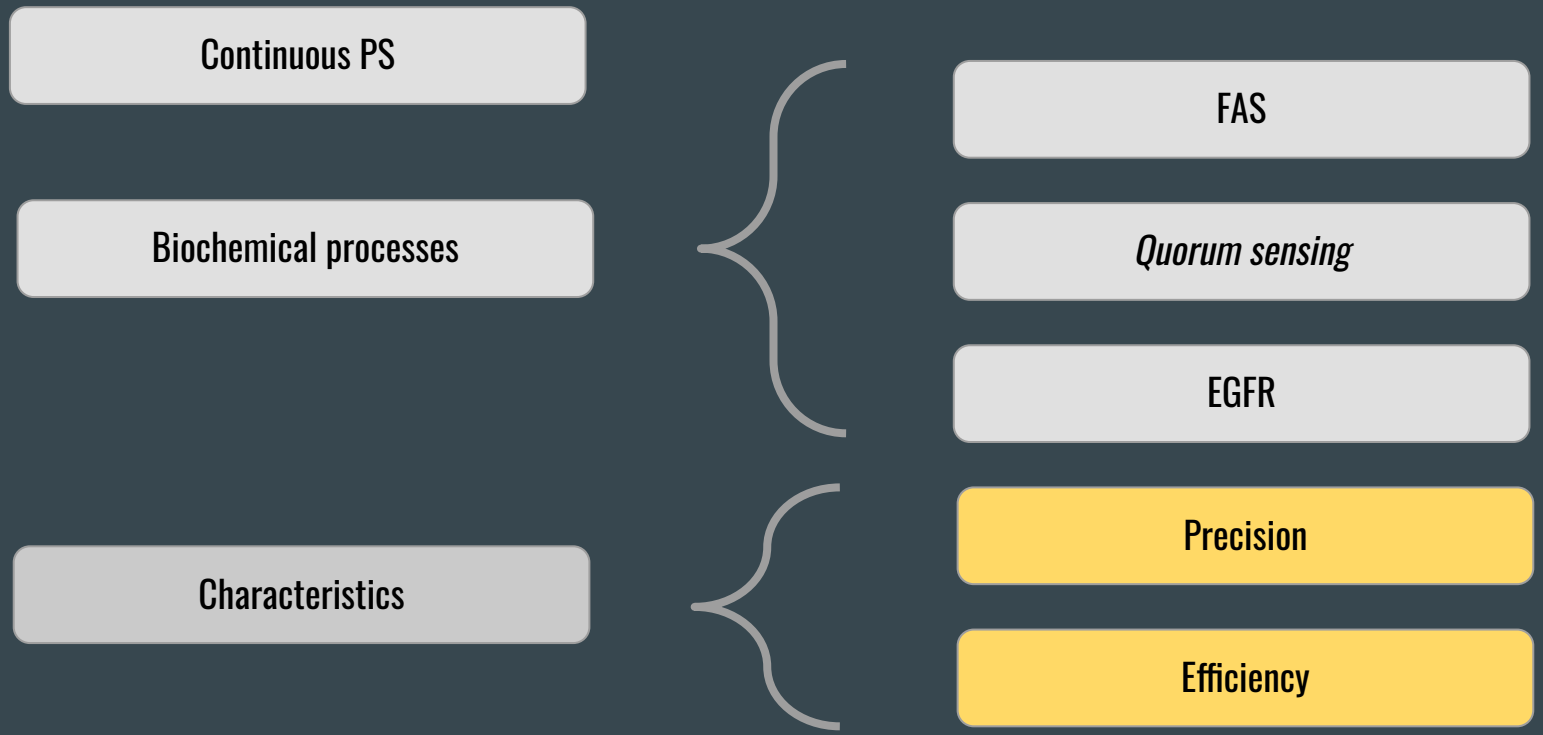
Classical Simulation Algorithms

Study cases



Classical Simulation Algorithms

Study cases



Classical Simulation Algorithms

The kernel of Gillespie's algorithm

Kernel of Gillespie's algorithm

Input

Initial state of a environment m

$r_1, ..., r_q$ rules with propensities p_j;

Execution

$p_0 = \sum_{j=1}^{q} p_j$;

$r_1 \leftarrow \mathcal{U}(0,1), r_2 \leftarrow \mathcal{U}(0,1)$

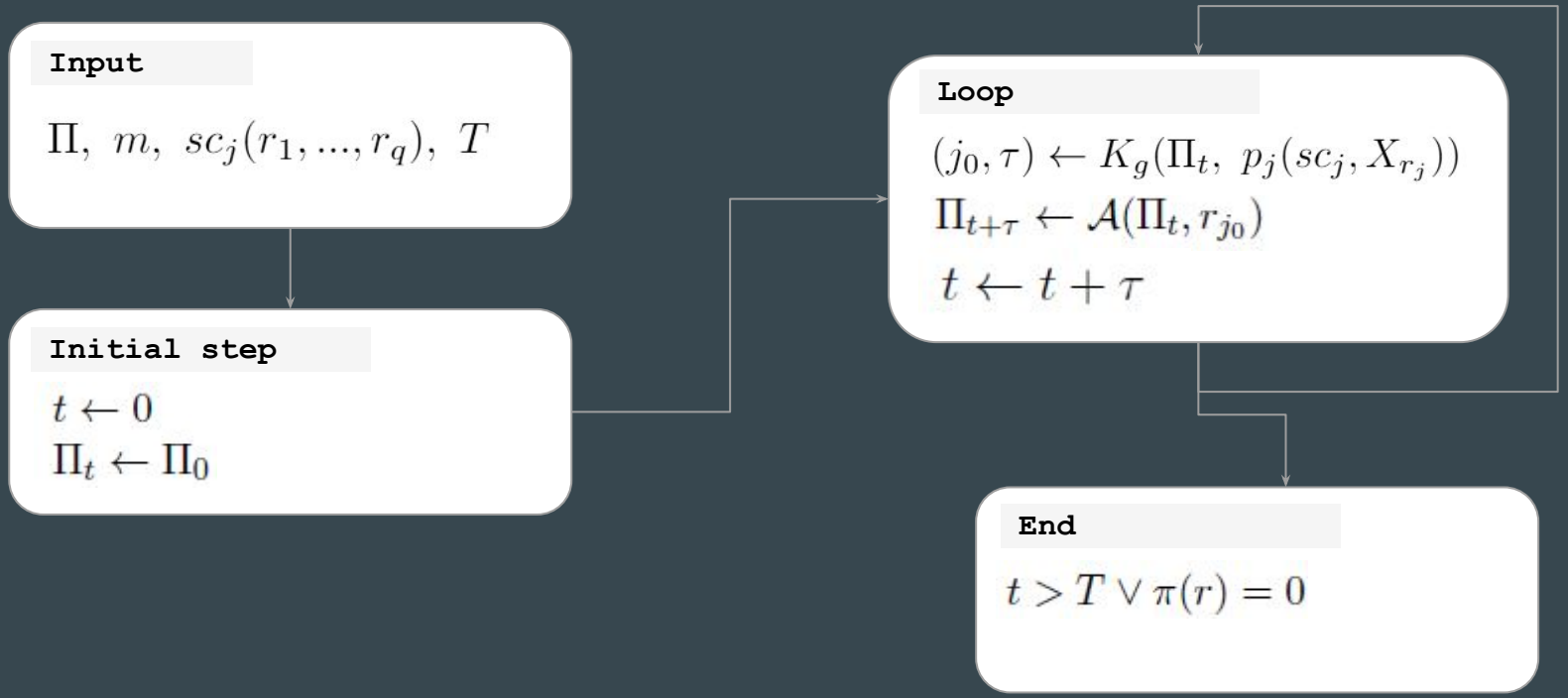
$j_0 \leftarrow j \geq 1$ s.t. $\sum_{k=1}^{j-1} p_k < r_1 \cdot p_0 \leq \sum_{k=1}^{j} p_k$

$\tau_i = \frac{1}{p_0} ln(\frac{1}{r_2})$

return (j_0, τ)

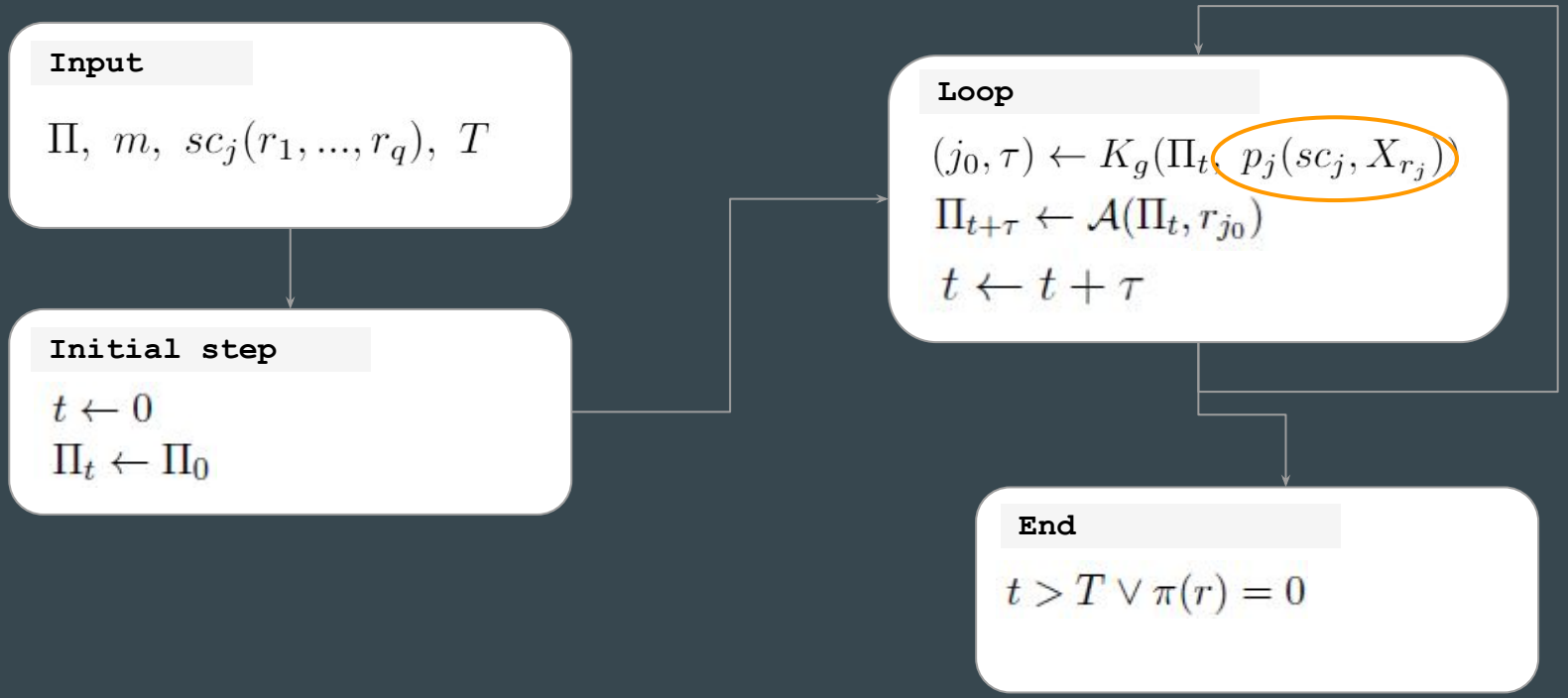
Classical Simulation Algorithms

Gillespie's algorithm



Classical Simulation Algorithms

Gillespie's algorithm



Classical Simulation Algorithms

Multicompartmental Gillespie's algorithm

Input

$\Pi,\ m,\ p_{j,i},\ j \in N_1^q, i \in \mathcal{C},\ T$

Initial step

$t \leftarrow 0$

$l \leftarrow \{(\tau_c, j_c, c) = \mathcal{G}(\Pi_c)\ :\ c \in \mathcal{C}\}$

Loop

$(\tau_m, j, m):\ \tau_m \leq \tau_c,\ \forall\ (\tau_c, j_c, c) \in l$

$\Pi_{t+\tau} \leftarrow \mathcal{A}(\Pi_t, r_j)$

$t \leftarrow t + \tau_m$

Actualiza $l, p_{j,i}$

End

$t > T \vee \pi(r) = 0$

Classical Simulation Algorithms

Multicompartmental Gillespie's algorithm

Input

$\Pi,\ m,\ p_{j,i},\ j \in N_1^q, i \in \mathcal{C},\ T$

Initial step

$t \leftarrow 0$

$l \leftarrow \{(\tau_c, j_c, c) = \mathcal{G}(\Pi_c)\ :\ c \in \mathcal{C}\}$

Loop

$(\tau_m, j, m):\ \tau_m \leq \tau_c,\ \forall\ (\tau_c, j_c, c) \in l$

$\Pi_{t+\tau} \leftarrow \mathcal{A}(\Pi_t, r_j)$

$t \leftarrow t + \tau_m$

Actualiza $l, p_{j,i}$

End

$t > T \vee \pi(r) = 0$

Classical Simulation Algorithms

Multicompartmental Gillespie's algorithm

Input

$\Pi,\ m,\ p_{j,i},\ j \in N_1^q, i \in \mathcal{C},\ T$

Initial step

$t \leftarrow 0$

$l \leftarrow \{(\tau_c, j_c, c) = \mathcal{G}(\Pi_c)\ :\ c \in \mathcal{C}\}$

Loop

$(\tau_m, j, m):\ \tau_m \leq \tau_c,\ \forall\ (\tau_c, j_c, c) \in l$

$\Pi_{t+\tau} \leftarrow \mathcal{A}(\Pi_t, r_j)$

$t \leftarrow t + \tau_m$

Actualiza $l, p_{j,i}$

End

$t > T \vee \pi(r) = 0$

Classical Simulation Algorithms

Deterministic waiting time algorithm

Loop

$(i_0, \tau_{i_0}, c_0) \leftarrow lista[0]$;

$t \leftarrow t + \tau_{i_0}$;

for $(i, \tau_i, c) \in lista$ **do**

$(i, \tau_i, c) \leftarrow (i, \tau_i - \tau_{i_0}, c)$;

end for

Apply rule r_{i_0} once

Delete (i_0, τ_{i_0}, c_0) from *lista*

for c' compartment affected by the rule

update (i, τ_i, c) related to compartment c'

Order by τ_i

end for

Classical Simulation Algorithms

Deterministic waiting time algorithm

Loop

$(i_0, \tau_{i_0}, c_0) \leftarrow lista[0]$;

$t \leftarrow t + \tau_{i_0}$;

for $(i, \tau_i, c) \in lista$ **do**

$(i, \tau_i, c) \leftarrow (i, \tau_i - \tau_{i_0}, c)$;

end for

Apply rule r_{i_0} once

Delete (i_0, τ_{i_0}, c_0) from *lista*

for c' compartment affected by the rule

update (i, τ_i, c) related to compartment c'

Order by τ_i

end for

$$\tau_{r_j} = p_j = k_j \cdot \prod_{i \in lhs} X_i$$

Classical Simulation Algorithms

Comparison

Multicompartmental Gillespie's Algorithm

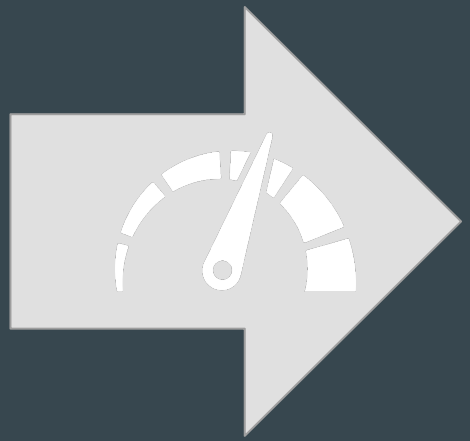
Deterministic Waiting Time Algorithm

The problem

The problem

Algorithms

Multicompartmental Gillespie's Algorithm



Deterministic Waiting Time Algorithm

The problem

Algorithms

Loop

$(i_0, \tau_{i_0}, c_0) \leftarrow lista[0]$;

$t \leftarrow t + \tau_{i_0}$;

for $(i, \tau_i, c) \in lista$ **do**

$(i, \tau_i, c) \leftarrow (i, \tau_i - \tau_{i_0}, c)$;

end for

Apply rule r_{i_0} once

Delete (i_0, τ_{i_0}, c_0) from *lista*

for c' compartment affected by the rule

update (i, τ_i, c) related to compartment c'

Order by τ_i

end for

The problem

Some math - FAS

$$r_{96} \equiv Bax[Bcl2]_m \rightarrow [Bcl2 : Bax]_m :: 2 \cdot 10^{-3}$$

$$[Bcl2]_m = 45172; \qquad [Bax]_c = 50189$$

$$\tau_{r_{96}} = \frac{1}{c_{r_{96}} \cdot P} = \frac{1}{2e{-}3} \cdot \frac{1}{[Bcl2]_m \cdot [Bax]_c} \sim 2 \cdot 10^{-7}$$

The problem

Some math - FAS

$$r_{96} \equiv Bax[Bcl2]_m \rightarrow [Bcl2 : Bax]_m :: 2 \cdot 10^{-3}$$

$$[Bcl2]_m = 45172; \qquad [Bax]_c = 50189$$

$$\tau_{r_{96}} = \frac{1}{c_{r_{96}} \cdot P} = \frac{1}{2e{-}3} \cdot \frac{1}{[Bcl2]_m \cdot [Bax]_c} \sim 2 \cdot 10^{-7}$$

The problem

Some math - FAS

$$\tau_{r_{96}} = \frac{1}{c_{r_{96}} \cdot P} = \frac{1}{2e{-}3} \cdot \frac{1}{[Bcl2]_m \cdot [Bax]_c} \sim 2 \cdot 10^{-7}$$

With 100 rules / second

* Only rule 96 is considered

The problem

Some math - FAS

$$\tau_{r_{96}} = \frac{1}{c_{r_{96}} \cdot P} = \frac{1}{2e{-}3} \cdot \frac{1}{[Bcl2]_m \cdot [Bax]_c} \sim 2 \cdot 10^{-7}$$

With 100 rules / second

1 second in the experiment ~* 1 day in real-life (!)

* Only rule 96 is considered

The problem

Some math - FAS

$$\tau_{r_{96}} = \frac{1}{c_{r_{96}} \cdot P} = \frac{1}{2e{-}3} \cdot \frac{1}{[Bcl2]_m \cdot [Bax]_c} \sim 2 \cdot 10^{-7}$$

With 100 rules / second

1 second in the experiment ~* 1 day in real-life (!)

8 h ~* 80 years (?!)

* Only rule 96 is considered

Ideas

Ideas

Propensities + lista

Loop

$(i_0, \tau_{i_0}, c_0) \leftarrow lista[0]$;

$t \leftarrow t + \tau_{i_0}$;

for $(i, \tau_i, c) \in lista$ **do**

$(i, \tau_i, c) \leftarrow (i, \tau_i - \tau_{i_0}, c)$;

end for

Apply rule r_{i_0} once

Delete (i_0, τ_{i_0}, c_0) from *lista*

for c' compartment affected by the rule

update (i, τ_i, c) related to compartment c'

Order by τ_i

end for

Update only a number of steps (propensities or the 3-tuples)

Right now...

Thanks

1. The used icons are from Uniconlabs, Freepik, Kiranshastry, mikan933, Iconjam and piksart.

Presenting *RAPS: R Aid for P Systems*

+

Simulating stochastic algorithms

•••

19th Brainstorming Week on Membrane Computing

José Antonio Rodríguez Gallego