Package `membranecomputing` (v0.2)

David Orellana-Martín
dorellana@us.es

September 23, 2021

Contents

1 Introduction

Membrane computing is a framework where different models of computations, called membrane systems or P systems arise from. The notation of these models is very variate since the number of researchers in the area is very high, and also their background is different, and therefore the notation can slightly change. The idea of this package is to cover all the possible variants in the area and their respective rules. Concerning the rules, the objective is twofold: On the one hand, to use commands to transcript a rule in a paper format; On the other hand, since P-Lingua is the *de facto* standard language for simulating P systems, there is a possibility to print the rules in the P-Lingua language, ready to be copied and pasted into a `.pli` file.

2 Package options

The `membranecomputing` package offers the option `blackboard`. The differences of this option can be found in Subsection 2.1.

2.1 Font option

Notation of P systems can change depending on the model. But even with the same model, some differences can be found between different papers. For instance, while some researchers use the symbol Γ to denote the working alphabet, other use the letter O. Other example is the use of the letters $\mathcal{M}_i$ or w_i for describing the initial multisets of the regions. For this purpose, this option has been implemented.

- `traditional` (*Default*) This typesets the symbols like $\Gamma, \Sigma, \mathcal{M}_i, \mathcal{R}_i, \rho_i$ [1].
- `blackboard` This typesets the symbols like O, E, w_i, R_i, p_i.

The rest of the symbols are defined without taking into account this option.

3 Using the package

In this section I will try to explain all the different uses of this package, from basic use to creation of new templates.

3.1 Basic notations

This is a list of some notations that can be used for defining initial multisets, sets of rules, and so on.

Working alphabet	`\wa`	Γ
Input alphabet	`\ia`	Σ
Labels set	`\ls`	H
Membrane structure	`\ms`	μ
Initial multiset	`\im{i}`	$\mathcal{M}_i$
Rule set	`\rs{i}`	$\mathcal{R}_i$
Probabilities set	`\ps{i}`	ρ_i
vE	`\vE`	v_E
Neuron	`\neuron{i}`	σ_i
Compartment	`\compartment{i}`	C_i
Agent	`\agent{i}`	B_i
Degree	`\degree`	q
Synapses	`\syn`	syn
Input region	`\iin`	i_{in}
Output region	`\iout`	i_{out}
Object yes	`\yes`	`yes`
Object no	`\no`	`no`

[1] For SNP systems, the singleton $\{a\}$, is still called O in this mode.

3.2 Languages and computability theory

Regular language	`\REG`	REG
Linear language	`\LIN`	LIN
Context-free language	`\CF`	CF
Context-sensitive language	`\CS`	CS
Recursively enumerable language	`\RE`	RE

To define a new set of languages, it is enough to make a new command as follows:

```
\newcommand{\L}{\compSet{L}}
```

This results in: L.

3.3 Families of membrane systems

Since the number of "ingredients" of the different variants of P systems is sometimes high, then some notations can be used to short them whenever they must be used. For instance, polarizationless P systems with active membranes when dissolution rules and division rules only for elementary membranes are allowed is usually denoted as $\mathcal{AM}^0(-d, +ne)$. To denote this, it is enough to write `\AM0{d, +ne}`. In this package, some examples of different families of P systems are defined.

Ps with am	`\AM[\alpha]{\beta}`	$\mathcal{AM}^{\alpha}(\beta)$
Polarizationless Ps with am	`\AM0{\alpha}`	$\mathcal{AM}^{0}(\alpha)$
Tissue Ps with s/a rules	`\TC[\alpha]{\beta}`	$\mathcal{T}\alpha\mathcal{C}(\beta)$
Tissue Ps with s/a and division rules	`\TDC{\alpha}`	$\mathcal{TDC}(\alpha)$
Tissue Ps with s/a and separation rules	`\TSC{\alpha}`	$\mathcal{TSC}(\alpha)$
Ps with s/a rules	`\CC[\alpha]{\beta}`	$\mathcal{C}\alpha\mathcal{C}(\beta)$
Ps with s/a and division rules	`\CDC{\alpha}`	$\mathcal{CDC}(\alpha)$
Ps with s/a and separation rules	`\CSC{\alpha}`	$\mathcal{CSC}(\alpha)$
Tissue Ps with evol. comm. rules	`\TEC[\alpha]{\beta}`	$\mathcal{T}\alpha\mathcal{EC}(\beta)$
Tissue Ps with evol. comm. and division rules	`\TDEC{\alpha}`	$\mathcal{TDEC}(\alpha)$
Tissue Ps with evol. comm. and separation rules	`\TSEC{\alpha}`	$\mathcal{TSEC}(\alpha)$
Ps with evol. comm. rules	`\CEC[\alpha]{\beta}`	$\mathcal{C}\alpha\mathcal{EC}(\beta)$
Ps with evol. comm. and division rules	`\CDEC{\alpha}`	$\mathcal{CDEC}(\alpha)$
Ps with evol. comm. and separation rules	`\CSEC{\alpha}`	$\mathcal{CSEC}(\alpha)$

To define a new notation for a family of membrane systems, it is enough to make a new command as follows:

```
\newcommand{\MS}[3]{\Pfamily{MS}{#1}{#2}{#3}}
```

This results in: $\mathcal{MS}^{\#2}_{\#3}(\#4)$.

3.4 Computational complexity theory

It is usual to define different complexity classes in the framework of membrane computing. In this sense, it would be interesting to automate the definition of these classes to avoid using all LaTeXcommands each time we want to use them.

`\PMC[\alpha]{\beta}`	$\mathbf{PMC}_{\beta}^{\alpha}$
`\PSPACEMC[\alpha]{\beta}`	$\mathbf{PSPACEMC}_{\beta}^{\alpha}$
`\EXPMC[\alpha]{\beta}`	$\mathbf{EXPMC}_{\beta}^{\alpha}$
`\EXPSPACEMC[\alpha]{\beta}`	$\mathbf{EXPPSPACEMC}_{\beta}^{\alpha}$

To define a new notation for a complexity class in membrane systems, it is enough to make a new command as follows:

```
\newcommand{\C}[3]{\complClass{C}{#1}{#2}}
```

This results in: $\mathbf{CMC}_{\#2}^{\#1}$ [2].

3.5 P systems

One of the two main contributions of this package is to make easier to name a P system. Usually, when we have to define a new membrane system, we have to write something like

```
\Pi = (\Gamma, \mu, H, \mathcal{M}_1, \mathcal{M}_{2}, \mathcal{R}, i_{out})
```

For making this task easier, the command `psystem` has been defined. Basically, it takes 5 arguments (first is optional) and can be used as follows: `\psystem[#1]{#2}{#3}{#4}{#5}`, where

- `#1` $\in \{$`recognizer`, `nonrecognizer`$\}$: adds an input alphabet and an input region.
- `#2` $\in \{$`cell`, `tissue`$\}$: adds a membrane structure in the first case.
- `#3` $\in \{$`transition`, `activemembranes`, `symportantiport`, `spiking`, `kernel`, `colony`$\}$: changes some of the parameters of the P system.
- `#4` is a subscript for the symbol Π.
- `#5` > 1 is the degree of the system.

For instance, the command

```
\psystem[recognizer]{cell}{activemembranes}{1}{10}
```

would lead to the following:

$$\Pi_1 = (\Gamma, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_{10}, \mathcal{R}_1, \ldots, \mathcal{R}_{10}, i_{in}, i_{out})$$

Some templates have been prepared for most of the usual variants of P systems:

[2] CMC does not stand for Conference on Membrane Computing here.

`\psystemAM`	$\Pi = (\Gamma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{out})$
`\rpsystemAM`	$\Pi = (\Gamma, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out})$
`\psystemSA`	$\Pi = (\Gamma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out})$
`\rpsystemSA`	$\Pi = (\Gamma, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$
`\SNpsystem`	$\Pi = (O, \sigma_1, \ldots, \sigma_q, syn, i_{out})$
`\rSNpsystem`	$\Pi = (O, \sigma_1, \ldots, \sigma_q, syn, i_{in}, i_{out})$
`\kpsystem`	$k\Pi = (\Gamma, \mu, C_1, \ldots, C_q, i_{out})$
`\rkpsystem`	$k\Pi = (\Gamma, \mu, C_1, \ldots, C_q, i_{in}, i_{out})$
`\pcolony`	$\Pi = (\Gamma, e, f, v_E, B_1, \ldots, B_q,)$
`\rpcolony`	$\Pi = (\Gamma, \Sigma, e, f, v_E, B_1, \ldots, B_q, i_{in},)$

All commands preceded by a `r` are the recognizer version of their r-less counterpart.

3.6 Rules

The (current) big work of this paper is to normalise the notation of the rules of P systems. Several differences can be found in the literature, most of them about spacing, position of elements, and so on. The idea is to unify these notations in one big element: the `\mcrule` command. This command tries to cover all types of rules in the membrane computing framework, including esoteric rules that could arise. The idea is to catch some basic elements and parse them in order to obtain the whole rule. As some types of P systems use different notations, as arrows, separators and so on, different cases have being contemplated. The definition of this command is `\mcrule[#1]{#2}{#3}{#4}{#5}{#6}`, where:

`#1` $\in$ {`written`(*Default*), `plingua`}: the first one makes it look as the usual scientific notation while the second one transforms the rule into P-Lingua notation.

`#2` It can be one of the following parameters:

`rewriting`	Classical rewriting rules	$u \to v$
`single`	Rules with a single pair of brackets	$[\, a \to b \,]_h$
`multiple`	Rules with different brackets in the LHS and the RHS	$[\, a \,]_h \to b\ [\ ]_h$
`paren`	Rules delimited by parentheses	$(i, u/v, j)$
`spike`	Spiking rules	$E/a^n \to a; d$

`#3` > 1: Number of regions in the LHS (the main region and the outside regions are counted as one)

`#4` > 1: Number of regions in the RHS (the main region and the outside regions are counted as one)

`#5` explained below

`#6` explained below

The `\mcrule` command is a tool capable of representing a wide variety of types of rule. The idea is to represent all the rules in a standardised way and well-spaced. An example with each type of rule is given:

- `rewriting`: `\mcrule{rewriting}{}{}{u}{v}` will produce $u \to v$.
- `single`: `\mcrule{single}{}{}{{a}{h}{+}}{{c_{2}\alpha}}` will produce $[\, a \to 2 \,]^{+}_{h}$.
- `multiple`: `\mcrule{multiple}{4}{4}{{{a}{3}{+}}{b}}{{e_{1}}{4}}{;}}{{}{{a}}{-}{{o}{!}}}` will produce $b\, [\, a\, [\, e_1 \,]_4 \,;\,]^{+}_{3} \to a\, -\, \{\, o \,\}$. I will explain in detail what means each thing:
  - `{4}{4}`: In the left-hand side of the rule, 4 "regions" will be analysed (in this case, membrane 3, its parent membrane, membrane 4 and a special symbol); In the right-hand side of the rule, 4 "regions" will be analysed (in this case, the empty "main" membrane, its parent membrane, a special symbol and a special region) [3].
  - `{{{a}{3}{+}}{b}}{{e_{1}}{4}}{;}}`: It is divided into four parts:
    - `{{a}{3}{+}}`: The "main" membrane of the rule has label 3 and polarisation +, and its contents is an object a.
    - `{{b}}`: The outer content of the "main" membrane is an object b.
    - `{{e_{1}}{4}}`: It has a inner membrane with label 4 and its contents is an object e_1.
    - `{;}`: It writes down a ; symbol [4].
  - `{{}{{a}}{-}{{o}{!}}}`: It is divided into four parts:
    - `{}`: The "main" membrane of the rule is not specified, so it is omitted [5].
    - `{{a}}`: The outer content of the "main" membrane is an object a.
    - `-` It writes down a $-$ symbol [6].
    - `{{o}{!}}`: The symbol ! in the label is a reserved character to write down a guard $\{o\}$ [7].
- `paren`: `\mcrule{paren}{}{}{{ab}{3}}{{c}{2}}` will produce $(3, ab/c, 2)$.
- `spike`: `\mcrule{spike}{}{}{{a^{3}}{a^{2}}}{{a}{1}}` will produce $a^3/a^2 \to a; 1$.

Using `plingua` instead of `written` as the optional parameter, it will write the rule in the P-Lingua syntax. Therefore, the `multiple` rule previously used

[3] It is a very strange rule, but it will be useful to explain some special cases to use in your own rules.

[4] Useful for creation/deletion rules in kP systems.

[5] Useful for dissolution rules.

[6] Also useful for creation/deletion rules in kP systems.

[7] Also used in kP systems.

in this section would render as
$\mathtt{b} + [\,\mathtt{a}\,[\,\mathtt{e}_1\,]'4\,;\,]'\,3\ --> \mathtt{a}\ -\ \{\,\mathtt{o}\,\}$.

With this, anyone can create almost any kind of rule that one could think [8]. However, for the sake of simplicity, I have defined some commands for the basic types of rules from the bibliography.

`\rewriting{u}{v}`	$u \to v$
`\rewritingT`	$u \to v$
`\evolution{a}{b}{h}{\alpha}`	$[\,a \to b\,]_h^{\alpha}$
`\evolutionT`	$[\,a \to b\,]_h^{\alpha}$
`\evolutionP{a}{b}{h}{\alpha}`	$\alpha\,[\,\mathtt{a}\ -->\ \mathtt{b}\,]'\mathtt{h}$
`\evolutionPT`	$\alpha\,[\,\mathtt{a}\ -->\ \mathtt{b}\,]'\mathtt{h}$
`\pevolution{a}{b}{h}`	$[\,a \to b\,]_h$
`\pevolutionT`	$[\,a \to b\,]_h$
`\pevolutionP{a}{b}{h}`	$[\,\mathtt{a}\ -->\ \mathtt{b}\,]'\mathtt{h}$
`\pevolutionPT`	$[\,\mathtt{a}\ -->\ \mathtt{b}\,]'\mathtt{h}$
`\antiport{u}{i}{v}{j}`	$(i, u/v, j)$
`\antiportT`	$(i, u/v, j)$
`\symportT`	$(i, u/\lambda, j)$
`\antiportP{u}{i}{v}{j}`	$(\mathtt{i}, \mathtt{u}/\mathtt{v}, \mathtt{j})$
`\antiportPT`	$(\mathtt{i}, \mathtt{u}/\mathtt{v}, \mathtt{j})$
`\symportPT`	$(\mathtt{i}, \mathtt{u}/\lambda, \mathtt{j})$
`\sendin{a}{b}{h}{\alpha_{1}}{\alpha_{2}}`	$a\ [\ ]_h^{\alpha_1} \to [\,b\,]_h^{\alpha_2}$
`\sendinT`	$a\ [\ ]_h^{\alpha_1} \to [\,b\,]_h^{\alpha_2}$
`\sendinP{a}{b}{h}{\alpha_{1}}{\alpha_{2}}`	$\mathtt{a}\,\alpha_1\,[\ ]'\mathtt{h}\ -->\ \alpha_2\,[\,\mathtt{b}\,]'\mathtt{h}$
`\sendinPT`	$\mathtt{a}\,\alpha_1\,[\ ]'\mathtt{h}\ -->\ \alpha_2\,[\,\mathtt{b}\,]'\mathtt{h}$
`\psendin{a}{b}{h}`	$a\ [\ ]_h \to [\,b\,]_h$
`\psendinT`	$a\ [\ ]_h \to [\,b\,]_h$
`\psendinP{a}{b}{h}`	$\mathtt{a}\ [\ ]'\mathtt{h}\ -->\ [\,\mathtt{b}\,]'\mathtt{h}$
`\psendinPT`	$\mathtt{a}\ [\ ]'\mathtt{h}\ -->\ [\,\mathtt{b}\,]'\mathtt{h}$
`\sendout{a}{b}{h}{\alpha_{1}}{\alpha_{2}}`	$[\,a\,]_h^{\alpha_1} \to b\ [\ ]_h^{\alpha_2}$
`\sendoutT`	$[\,a\,]_h^{\alpha_1} \to b\ [\ ]_h^{\alpha_2}$
`\sendoutP{a}{b}{h}{\alpha_{1}}{\alpha_{2}}`	$\alpha_1\,[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}\,\alpha_2\,[\ ]'\mathtt{h}$
`\sendoutPT`	$[\,a\,]_h^{\alpha_1} \to b\ [\ ]_h^{\alpha_2}$
`\psendout{a}{b}{h}`	$[\,a\,]_h \to b\ [\ ]_h$
`\psendoutT`	$[\,a\,]_h \to b\ [\ ]_h$
`\psendoutP{a}{b}{h}`	$[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}\ [\ ]'\mathtt{h}$
`\psendoutPT`	$[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}\ [\ ]'\mathtt{h}$
`\dissolution{a}{b}{h}{\alpha}`	$[\,a\,]_h^{\alpha} \to b$
`\dissolutionT`	$[\,a\,]_h^{\alpha} \to b$
`\dissolutionP{a}{b}{h}{\alpha}`	$\alpha[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}$
`\dissolutionPT`	$\alpha[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}$
`\pdissolution{a}{b}{h}`	$[\,a\,]_h \to b$
`\pdissolutionT`	$[\,a\,]_h \to b$
`\pdissolutionP{a}{b}{h}`	$[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}$
`\pdissolutionPT`	$[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \mathtt{b}$
`\division{a}{b}{c}{h}{\alpha}{\alpha_{1}}{\alpha_{2}}`	$[\,a\,]_h^{\alpha} \to [\,b\,]_h^{\alpha_1}\,[\,c\,]_h^{\alpha_2}$
`\divisionT`	$[\,a\,]_h^{\alpha} \to [\,b\,]_h^{\alpha_1}\,[\,c\,]_h^{\alpha_2}$
`\divisionP{a}{b}{c}{h}{\alpha}{\alpha_{1}}{\alpha_{2}}`	$\alpha\,[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \alpha_1[\,\mathtt{b}\,]'\mathtt{h}\,\alpha_2[\,\mathtt{c}\,]'\mathtt{h}$
`\divisionPT`	$\alpha\,[\,\mathtt{a}\,]'\mathtt{h}\ -->\ \alpha_1[\,\mathtt{b}\,]'\mathtt{h}\,\alpha_2[\,\mathtt{c}\,]'\mathtt{h}$
`\pdivision{a}{b}{c}{h}`	$[\,a\,]_h \to [\,b\,]_h\,[\,c\,]_h$
`\pdivisionT`	$[\,a\,]_h \to [\,b\,]_h\,[\,c\,]_h$
`\pdivisionP{a}{b}{c}{h}`	$[\,\mathtt{a}\,]'\mathtt{c}\ -->\ \mathtt{h}[\,\mathtt{b}\,]'\mathtt{c}\,[\ ]'\mathtt{c}$

[8] Do you have any idea about other kind of rule? Please write me so I can think in a way to implement it.

`\pdivisionPT`	$[\mathtt{a}]'\mathtt{b} \dashrightarrow \mathtt{c}[\mathtt{h}]'\mathtt{b}[\,]'\mathtt{b}$
`\separation{a}{h}{\alpha}{\alpha_{1}}{\alpha_{2}}`	$[a]_h^{\alpha} \to [\Gamma_0]_h^{\alpha_1}[\Gamma_1]_h^{\alpha_2}$
`\separationT`	$[a]_h^{\alpha} \to [\Gamma_0]_h^{\alpha_1}[\Gamma_1]_h^{\alpha_2}$
`\separationP{a}{h}{\alpha}{\alpha_{1}}{\alpha_{2}}`	$\alpha[\mathtt{a}]'\mathtt{h} \dashrightarrow \alpha_1[\Gamma_0]'\mathtt{h}\,\alpha_2[\Gamma_1]'\mathtt{h}$
`\separationPT`	$\alpha[\mathtt{a}]'\mathtt{h} \dashrightarrow \alpha_1[\Gamma_0]'\mathtt{h}\,\alpha_2[\Gamma_1]'\mathtt{h}$
`\pseparation{a}{h}`	$[a]_h \to [\Gamma_0]_h[\Gamma_1]_h$
`\pseparationT`	$[a]_h \to [\Gamma_0]_h[\Gamma_1]_h$
`\pseparationP{a}{h}`	$[\mathtt{a}]'\mathtt{h} \dashrightarrow [\Gamma_0]'\mathtt{h}[\Gamma_1]'\mathtt{h}$
`\pseparationPT`	$[\mathtt{a}]'\mathtt{h} \dashrightarrow [\Gamma_0]'\mathtt{h}[\Gamma_1]'\mathtt{h}$
`\creation{a}{b}{c}{h}{h_{1}}{\alpha}{\alpha_{1}}{\alpha_{2}}`	$[a]_h^{\alpha} \to \left[b\,[c]_{h_1}^{\alpha_2}\right]_h^{\alpha_1}$
`\creationT`	$[a]_h^{\alpha} \to \left[b\,[c]_{h_1}^{\alpha_2}\right]_h^{\alpha_1}$
`\creationP{a}{b}{c}{h}{h_{1}}{\alpha}{\alpha_{1}}{\alpha_{2}}`	$\alpha[\mathtt{a}]'\mathtt{h} \dashrightarrow \alpha_1[\mathtt{b}\,\alpha_2[\mathtt{c}]'\mathtt{h}_1]'\mathtt{h}$
`\creationPT`	$\alpha[\mathtt{a}]'\mathtt{h} \dashrightarrow \alpha_1[\mathtt{b}\,\alpha_2[\mathtt{c}]'\mathtt{h}_1]'\mathtt{h}$
`\pcreation{a}{b}{c}{h}{h_{1}}`	$[a]_h \to \left[b\,[c]_{h_1}\right]_h$
`\pcreationT`	$[a]_h \to \left[b\,[c]_{h_1}\right]_h$
`\pcreationP{a}{b}{c}{h}{h_{1}}`	$[\mathtt{a}]'\mathtt{h} \dashrightarrow [\mathtt{b}[\mathtt{c}]'\mathtt{h}_1]'\mathtt{h}$
`\pcreationPT`	$[\mathtt{a}]'\mathtt{h} \dashrightarrow [\mathtt{b}[\mathtt{c}]'\mathtt{h}_1]'\mathtt{h}$
`\spiking{E}{a^{n}}{a}{d}`	$E/a^n \to a; d$
`\spikingT`	$E/a^n \to a; d$
`\forgettingT`	$a^n \to \lambda;$
`\spikingP{E}{a^{n}}{a}{d}`	$E/a^n \to a; d$
`\spikingPT`	$E/a^n \to a; d$
`\forgettingPT`	$a^n \to \lambda;$
`\krewriting{x}{y}{g}`	$x \to y\,\{g\}$
`\krewritingT`	$x \to y\,\{g\}$
`\krewritingP{x}{y}{g}`	$\mathtt{x} \dashrightarrow \mathtt{y}\,\{\mathtt{g}\}$
`\krewritingPT`	$\mathtt{x} \dashrightarrow \mathtt{y}\,\{\mathtt{g}\}$
`\linkcreation{x}{y}{t_{l_{i}}}{t_{l_{j}}}{g}`	$[x]_{t_{l_i}}\,;[\,]_{t_{l_j}} \to [y]_{t_{l_i}} - [\,]_{t_{l_j}}\,\{g\}$
`\linkcreationT`	$[x]_{t_{l_i}}\,;[\,]_{t_{l_j}} \to [y]_{t_{l_i}} - [\,]_{t_{l_j}}\,\{g\}$
`\linkcreationP{x}{y}{t_{l_{i}}}{t_{l_{j}}}{g}`	$[\mathtt{x}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}}\,;[\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}} \dashrightarrow [\mathtt{y}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}} - [\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}}\,\{\mathtt{g}\}$
`\linkcreationPT`	$[\mathtt{x}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}}\,;[\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}} \dashrightarrow [\mathtt{y}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}} - [\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}}\,\{\mathtt{g}\}$
`\linkdestruction{x}{y}{t_{l_{i}}}{t_{l_{j}}}{g}`	$[x]_{t_{l_i}} - [\,]_{t_{l_j}} \to [y]_{t_{l_i}}\,;[\,]_{t_{l_j}}\,\{g\}$
`\linkdestructionT`	$[x]_{t_{l_i}} - [\,]_{t_{l_j}} \to [y]_{t_{l_i}}\,;[\,]_{t_{l_j}}\,\{g\}$
`\linkdestructionP{x}{y}{t_{l_{i}}}{t_{l_{j}}}{g}`	$[\mathtt{x}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}} - [\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}} \dashrightarrow [\mathtt{y}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}}\,;[\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}}\,\{\mathtt{g}\}$
`\linkdestructionPT`	$[\mathtt{x}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}} - [\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}} \dashrightarrow [\mathtt{y}]'\mathtt{t}_{\mathtt{l}_\mathtt{i}}\,;[\,]'\mathtt{t}_{\mathtt{l}_\mathtt{j}}\,\{\mathtt{g}\}$
`\tissueevolcomm{u}{v}{v'}{u'}{i}{j}`	$[u]_i[v]_j \to [v']_i[u']_j$
`\tissueevolcommT`	$[u]_i[v]_j \to [v']_i[u']_j$
`\tissueevolsympT`	$[u]_i[\,]_j \to [\,]_i[u']_j$
`\tissueevolcommP{u}{v}{v'}{u'}{i}{j}`	$[\mathtt{u}]'\mathtt{i}[\mathtt{v}]'\mathtt{j} \dashrightarrow [\mathtt{v}']'\mathtt{i}[\mathtt{u}']'\mathtt{j}$
`\tissueevolcommPT`	$[\mathtt{u}]'\mathtt{i}[\mathtt{v}]'\mathtt{j} \dashrightarrow [\mathtt{v}']'\mathtt{i}[\mathtt{u}']'\mathtt{j}$
`\tissueevolsympPT`	$[\mathtt{u}]'\mathtt{i}[\,]'\mathtt{j} \dashrightarrow [\,]'\mathtt{i}[\mathtt{u}']'\mathtt{j}$
`\evolcomm{u}{v}{v'}{u'}{i}{j}`	$\left[u\,[v]_j\right]_i \to \left[v'\,[u']_j\right]_i$
`\evolcommT`	$\left[u\,[v]_j\right]_i \to \left[v'\,[u']_j\right]_i$
`\evolsyminT`	$\left[u\,[\,]_j\right]_i \to \left[\,[u']_j\right]_i$
`\evolsymoutT`	$\left[\,[u]_j\right]_i \to \left[u'\,[\,]_j\right]_i$
`\evolcommP{u}{v}{v'}{u'}{i}{j}`	$[\mathtt{u}[\mathtt{v}]'\mathtt{j}]'\mathtt{i} \dashrightarrow [\mathtt{v}'[\mathtt{u}']'\mathtt{j}]'\mathtt{i}$
`\evolcommPT`	$[\mathtt{u}[\mathtt{v}]'\mathtt{j}]'\mathtt{i} \dashrightarrow [\mathtt{v}'[\mathtt{u}']'\mathtt{j}]'\mathtt{i}$
`\evolsyminPT`	$[\mathtt{u}[\,]'\mathtt{j}]'\mathtt{i} \dashrightarrow [\,[\mathtt{u}']'\mathtt{j}]'\mathtt{i}$
`\evolsymoutPT`	$[\,[\mathtt{u}]'\mathtt{j}]'\mathtt{i} \dashrightarrow [\mathtt{u}'[\,]'\mathtt{j}]'\mathtt{i}$

As you can see, the names follow a pattern:

- Their name is descriptive in terms of the rule it represents.
- If the rule starts with a `p`, it means that it is a polarizationless rule.
- If the rule ends with (or its second to last letter is) a `P`, it means it will write the rule in P-Lingua format.
- If the rule ends with a `T`, it means it is a rule template; that is, the typical rule used to describe its behaviour.

If you have an idea to make the notation simpler, please let me know.

4 Future work

A TODO list for the future development of the package. Please do not hesitate to contact me if you think that something should be included in this list, or if there is any bug or concept that should be included as soon as possible. Please, contact me at dorellana@us.es

- Add new variants of P systems, as well as some new templates for other rules.
- Clean the code and create new command for analysing the `multiple` case automatically.
- Add the possibility of define rules with more than one level of deepness.
- Add a automatic parsing of a membrane structure, so a well-spaced structure can appear in a paper. It would be interesting to also let it export a string in the P-Lingua format.
- A long-term objective is to export a .pdf with the picture of a P system with its contents. Creating an image for your paper could be as simple as defining it as you usually do, it would be cool, right?

I hope you find the package interesting and useful for your purposes. And, as I said above, please do not hesitate to contact me if you have any questions of or suggestions for it.