Package ‘petrinetR’

March 8, 2019

Type Package
Title Building, Visualizing, Exporting and Replaying Petri Nets
Version 0.2.1
Date 2019-03-08
Description Functions for the construction of Petri Nets. Petri Nets can be replayed by firing enabled transitions. Silent transitions will be hidden by the execution handler. Also includes functionalities for the visualization of Petri Nets and export of Petri Nets to PNML (Petri Net Markup Language) files.
License GPL-3
Depends R(>= 3.0.0)
LazyData true
Imports dplyr, visNetwork, DiagrammeR, xml2, purrr
RoxygenNote 6.1.1
URL https://www.bupar.net
BugReports https://github.com/gertjanssenswillen/petrinetR/issues
NeedsCompilation no
Author Gert Janssenswillen [aut, cre]
Maintainer Gert Janssenswillen <gert.janssenswillen@uhasselt.be>
Repository CRAN
Date/Publication 2019-03-08 11:30:03 UTC

R topics documented:

create_PN .......................................................... 2
enabled ............................................................ 3
enabled_transition ................................................ 3
execute ............................................................. 4
flows ................................................................. 4
is_place ............................................................. 5
create_PN

Description

Function to create a petri net by specifying a set of places, transitions, flows and a marking.

Usage

create_PN(places, transitions, flows, marking)

Arguments

places A vector of unique places.
transitions A vector of unique transitions.
flows A data.frame of flows, with columns named "from" and "to".
marking The names of the places to be marked.

Examples

create_PN("place_1",
"transition_1",
data.frame(from = "place_1", to = "transition_1"),
marking = "place_1")
**enabled**

### Description

List the enabled transitions in a marked Petri Net. Silent transitions, i.e. starting with "inv_" or "tau" are assumed to be able to fire silently, thereby possible enabling other transitions.

### Usage

```plaintext
enabled(PN)
```

### Arguments

- **PN**: A Petri Net

---

**enabled_transition**

### Description

Check if a transition is currently enabled

### Usage

```plaintext
enabled_transition(PN, transition)
```

### Arguments

- **PN**: A Petri Net
- **transition**: A Transition
**execute**

*Execute*

**Description**

Executes (fire) an enabled transition and returns the Petri Net with the New marking. If the transition is enabled via the firing of silent transition (i.e. starting with "inv_" of "tau"), it will fire these first. If the transition is not enabled, it will return FALSE.

**Usage**

```plaintext
execute(PN, transition)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>A Petri Net</td>
</tr>
<tr>
<td>transition</td>
<td>The transition to be fired</td>
</tr>
</tbody>
</table>

**flows**

*Flows*

**Description**

Extracts the flows from a Petri Net

**Usage**

```plaintext
flows(PN)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>A Petri Net</td>
</tr>
</tbody>
</table>
### is_place

**Description**
Check if a place is part of a petri net

**Usage**
\[
\text{is\_place} (\text{place}, \text{PN})
\]

**Arguments**
- **place**: A place
- **PN**: A Petri Net

### is_transition

**Description**
Check if a transition is part of a petri net

**Usage**
\[
\text{is\_transition} (\text{transition}, \text{PN})
\]

**Arguments**
- **transition**: A transition
- **PN**: A Petri Net

### marking

**Description**
Get the current marking of a Petri Net

**Usage**
\[
\text{marking} (\text{PN})
\]

**Arguments**
- **PN**: A Petri Net
Description

Several auxiliary functions for Petri Net objects.

Usage

n_places(pn)

n_transitions(pn)

n_flows(pn)

n_nodes(pn)

nodes(pn)

rename_transitions(pn, .f, ...)

rename_places(pn, .f, ...)

add_places(pn, .p)

add_transitions(pn, .t)

add_flows(pn, .flows)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>A petri net</td>
</tr>
<tr>
<td>.f</td>
<td>A function name to apply for renaming</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments</td>
</tr>
<tr>
<td>.p</td>
<td>A character vector of places</td>
</tr>
<tr>
<td>.t</td>
<td>A character vector of transitions</td>
</tr>
<tr>
<td>.flows</td>
<td>A data.frame with a to and from column</td>
</tr>
</tbody>
</table>
parsel_trace  Parse (logical)

**Description**
Tests whether a sequence of transitions can be fired by a Petri Net. If so returns TRUE, otherwise FALSE.

**Usage**
```
parsel_trace(PN, trace)
```

**Arguments**
- **PN**: A Petri Net
- **trace**: A sequence of transitions, stored in a vector.

**Parse**

**Description**
 Parses a sequence of transitions. If possible returns the Petri Net with the updated marking. Otherwise returns FALSE.

**Usage**
```
parse_trace(PN, trace)
```

**Arguments**
- **PN**: A Petri Net
- **trace**: A sequence of transitions, stored in a vector.
part_of

**Part of**

**Description**
Check if a node is part of a petri net

**Usage**
part_of(node, PN)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>A node</td>
</tr>
<tr>
<td>PN</td>
<td>A Petri Net</td>
</tr>
</tbody>
</table>

**petrinetR**

*petrinetR - Building, visualizing, exporting and replaying Petri Nets*

**Description**
Functions for the construction of Petri Nets. Petri Nets can be replayed by firing enabled transitions. Silent transitions will be hidden by the execution handler. Also includes functionalities for the visualization of Petri Nets and export of Petri Nets to PNML-files.

**places**

**Places**

**Description**
Extracts the places from a Petri Net

**Usage**
places(PN)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PN</td>
<td>A Petri Net</td>
</tr>
</tbody>
</table>
\textbf{post\textunderscore set} \hspace{1cm} Postset

\textbf{Description}

Get the postset of a transition or place in a Petri Net

\textbf{Usage}

\texttt{post\textunderscore set}(\texttt{PN}, \texttt{node})

\textbf{Arguments}

\begin{tabular}{ll}
\texttt{PN} & A Petri Net \\
\texttt{node} & A place or transition in the petri net
\end{tabular}

\textbf{pre\textunderscore set} \hspace{1cm} Preset

\textbf{Description}

Get the preset of a transition or place in a Petri Net

\textbf{Usage}

\texttt{pre\textunderscore set}(\texttt{PN}, \texttt{node})

\textbf{Arguments}

\begin{tabular}{ll}
\texttt{PN} & A Petri Net \\
\texttt{node} & A place or transition in the petri net
\end{tabular}
### print.petrinet

*Generic print function for petrinet*

**Description**

Generic print function for petrinet

**Usage**

```r
## S3 method for class 'petrinet'
print(x, ...)
```

**Arguments**

- `x`: petrinet object
- `...`: Additional Arguments

---

### read_pn

*Read PNML*

**Description**

Function

**Usage**

```r
read_pn(file)
```

**Arguments**

- `file`: Path to .pnml file

---

### render_pn

*Render Petri Net*

**Description**

Function

**Usage**

```r
render_pn(PN)
```

**Arguments**

- `PN`: A petri net
transitions

Description
Extracts the transitions from a Petri Net

Usage
transitions(PN)

Arguments
PN A Petri Net

tree_to_PN

Description
Create of petri net from a process tree.

Usage
tree_to_PN(tree)

Arguments
tree The process tree to be converted

visNetwork_from_PN

Description
Visualize a Petri Net with an interactive network

Usage
visNetwork_from_PN(PN)

Arguments
PN Petri Net to visualize
## Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_flows (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>add_places (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>add_transitions (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>create_PN</td>
<td>2</td>
</tr>
<tr>
<td>enabled, (n_places)</td>
<td>3</td>
</tr>
<tr>
<td>enabled_transition, (n_places)</td>
<td>3</td>
</tr>
<tr>
<td>execute, (n_places)</td>
<td>4</td>
</tr>
<tr>
<td>flows, (n_places)</td>
<td>4</td>
</tr>
<tr>
<td>is_place, (n_places)</td>
<td>5</td>
</tr>
<tr>
<td>is_transition, (n_places)</td>
<td>5</td>
</tr>
<tr>
<td>marking, (n_places)</td>
<td>5</td>
</tr>
<tr>
<td>n_flows (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>n_nodes (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>n_places, (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>n_transitions (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>nodes (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>parse_trace</td>
<td>7</td>
</tr>
<tr>
<td>parsel_trace</td>
<td>7</td>
</tr>
<tr>
<td>part_of, (n_places)</td>
<td>8</td>
</tr>
<tr>
<td>petrinetR, (n_places)</td>
<td>8</td>
</tr>
<tr>
<td>petrinetR-package (petrinetR), (n_places)</td>
<td>8</td>
</tr>
<tr>
<td>places, (n_places)</td>
<td>8</td>
</tr>
<tr>
<td>post_set, (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>pre_set, (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>print.petrinet, (n_places)</td>
<td>10</td>
</tr>
<tr>
<td>read_PN, (n_places)</td>
<td>10</td>
</tr>
<tr>
<td>rename_places (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>rename_transitions (n_places), (n_places)</td>
<td>6</td>
</tr>
<tr>
<td>render_PN</td>
<td>10</td>
</tr>
<tr>
<td>transitions, (n_places)</td>
<td>11</td>
</tr>
<tr>
<td>tree_to_PN, (n_places)</td>
<td>11</td>
</tr>
<tr>
<td>visNetwork_from_PN</td>
<td>11</td>
</tr>
</tbody>
</table>