Package ‘petrinetR’

April 3, 2023

Type Package
Title Building, Visualizing, Exporting and Replaying Petri Nets
Version 0.3.0
Date 2023-04-01
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)
Imports dplyr, visNetwork, DiagrammeR, xml2, purrr, lifecycle
RoxygenNote 7.2.3
Encoding UTF-8
BugReports https://github.com/bupavere/petrinetR/issues
NeedsCompilation no
Author Gert Janssenswillen [aut, cre], Ivan Esin [ctb]
Maintainer Gert Janssenswillen <gert.janssenswillen@uhasselt.be>
Repository CRAN
Date/Publication 2023-04-03 13:20:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_marked_PN</td>
<td>2</td>
</tr>
<tr>
<td>create_PN</td>
<td>3</td>
</tr>
<tr>
<td>enabled</td>
<td>3</td>
</tr>
<tr>
<td>enabled_transition</td>
<td>4</td>
</tr>
<tr>
<td>execute</td>
<td>4</td>
</tr>
</tbody>
</table>
create_marked_PN

Description

Function to create a marked_petrinet, consisting of a petrinet, an initial marking, and a final marking.

Usage

create_marked_PN(PN, initial_marking, final_marking)

Arguments

PN petrinet: Object of class petrinet.
initial_marking character: A vector with place ids representing the initial marking.
final_marking character: A vector with place ids representing the final marking.

Value

A marked_petrinet
create_PN

Create Petri Net

Description

Function to create a petrinet by specifying places, transitions and flows.

Usage

create_PN(places, transitions, flows)

Arguments

places data.frame or tibble of places, with columns id and label. Both columns should be characters.
transitions data.frame or tibble of transitions, with columns id and label. Both columns should be characters.
flows data.frame or tibble of flows, with columns named "from" and "to", referring to ids of places and transitions. Both columns should be characters.

Value

A petrinet

Examples

library(dplyr)
create_PN(tibble(id = "p1", label = "place_1"),
tibble(id = "t1", label = "transition_1"),
tibble(from = "t1", to = "p1"))

enabled

Enabled transitions

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

enabled(PN)

Arguments

PN A Petri Net
### enabled_transition  
**Enabled Transition**

**Description**
Check if a transition is currently enabled

**Usage**
```
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**
```
execute(PN, transition)
```

**Arguments**
- **PN**  
  A Petri Net
- **transition**  
  The transition to be fired
**final_marking**

**Final Marking**

**Description**
Get the final marking of a `marked_petrinet`

**Usage**
`final_marking(PN)`

**Arguments**
- **PN**
  - A `marked_petrinet`

---

**flows**

**Flows**

**Description**
Extracts the flows from a (marked) Petri Net

**Usage**
`flows(PN)`

```r
## S3 method for class 'petrinet'
flows(PN)
```

```r
## S3 method for class 'marked_petrinet'
flows(PN)
```

**Arguments**
- **PN**
  - petrinet or marked_petrinet

**Value**
A data.frame containing the flows of the petri net.

**Methods (by class)**
- `flows(petrinet)`: Flow of petrinet
- `flows(marked_petrinet)`: Flow of marked petrinet
### initial_marking \( \text{Initial Marking} \)

**Description**

Get the initial marking of a `marked_petrinet`.

**Usage**

```
initial_marking(PN)
```

**Arguments**

- **PN**: A `marked_petrinet`.

### is_node \( \text{Is node} \)

**Description**

Check if a node is part of a petri net.

**Usage**

```
is_node(node, PN)
```

**Arguments**

- **node**: character of length one: the node id to check.
- **PN**: petri net or `marked_petrinet`.

**Value**

Logical that indicates whether node is a node in PN.
is_place

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

Usage
is_place(place, PN)

Arguments
place character of length one: the place id to check.
PN petrinet or marked_petrinet

Value
logical that indicates whether place is a place in PN

is_transition

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

Usage
is_transition(transition, PN)

Arguments
transition character of length one: the transition id to check.
PN petrinet or marked_petrinet

Value
logical that indicates whether transition is a transition in PN
marked_petrinet

Marked petrinet

Description
Object consisting of a petrinet, initial marking, and final marking

marking

Marking

Description
Get the current marking of a Petri Net

Usage
marking(PN)

Arguments
PN A Petri Net

nodes

Get nodes from (marked) petrinet

Description
Get nodes from (marked) petrinet

Usage
nodes(PN)

Arguments
PN petrinet or marked_petrinet
Description

Several auxiliary functions for Petri Net objects.

Usage

n_places(PN)

n_transitions(PN)

n_flows(PN)

n_nodes(PN)

rename_transitions(PN, .f, ...)

rename_places(PN, .f, ...)

add_places(PN, places)

add_transitions(PN, transitions)

add_flows(PN, flows)

Arguments

PN         A petri net
.f         A function name to apply for renaming
...        Additional arguments
places     data.frame or tibble of places, with columns id and label. Both columns should be characters.
transitions data.frame or tibble of transitions, with columns id and label. Both columns should be characters.
flows      data.frame or tibble of flows, with columns named "from" and "to", referring to ids of places and transitions. Both columns should be characters.
parse_trace  Parse (logical)

Description

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

Usage

parse_trace(PN, trace)

Arguments

PN  A Petri Net
trace  A sequence of transitions, stored in a vector.

parse_trace  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.
Description

Check if a node is part of a petri net

Usage

`part_of(node, PN)`

Arguments

- `node`: A node
- `PN`: A Petri Net

petrinet

Object consisting of places, transitions and flows that denote a petri net

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**
- **PN**  
  • petrinet or marked_petrinet

**Methods (by class)**
- places(petrinet): Places of petrinet
- places(marked_petrinet): Places of marked petrinet

---

**post_set**  
*Postset*

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

**Usage**
```
post_set(PN, node)
```

**Arguments**
- **PN**  
  • petrinet or marked_petrinet
- **node**  
  • character of length one: the node id for which to get the postset.
**pre_set**

<table>
<thead>
<tr>
<th>Preset</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

```
pre_set(PN, node)
```

**Arguments**

- **PN**
  - `petrinet` or `marked_petrinet`
- **node**
  - `character` of length one: the node id for which to get the postset.

---

**read_PN**

| Read .PNML file |

**Description**

Read .PNML file

**Usage**

```
read_PN(file, add_final_marking = TRUE)
```

**Arguments**

- **file**
  - Path to .pnml file
- **add_final_marking**
  - `logical` (default: TRUE): add final marking. If TRUE, all places without outgoing flows are considered part of a single final marking. Overwrite with `set_final_marking()` if needed. If FALSE, final_marking is set to NULL

**Value**

- A `marked_petrinet`
render_PN  Render Petri Net

**Description**
Visualize Petri Net with bipartite graph.

**Usage**
render_PN(PN)

**Arguments**
- PN  petrinet or marked_petrinet

transitions  Transitions

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

**Usage**
transitions(PN)

```r
## S3 method for class 'petrinet'
transitions(PN)

## S3 method for class 'marked_petrinet'
transitions(PN)
```

**Arguments**
- PN  petrinet or marked_petrinet

**Methods (by class)**
- transitions(petrinet): Transitions of petrinet
- transitions(marked_petrinet): Transitions of marked petrinet
visNetwork_from_PN

Description

Visualize a Petri Net with an interactive network

Usage

visNetwork_from_PN(PN)

Arguments

PN petrinet or marked_petrinet
# Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_flows (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>add_places (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>add_transitions (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>character</td>
<td>2, 6, 7, 12, 13</td>
</tr>
<tr>
<td>create_marked_PN</td>
<td>2</td>
</tr>
<tr>
<td>create_PN</td>
<td>3</td>
</tr>
<tr>
<td>enabled</td>
<td>3</td>
</tr>
<tr>
<td>enabled_transition</td>
<td>4</td>
</tr>
<tr>
<td>execute</td>
<td>4</td>
</tr>
<tr>
<td>final_marking</td>
<td>5</td>
</tr>
<tr>
<td>flows</td>
<td>5</td>
</tr>
<tr>
<td>initial_marking</td>
<td>6</td>
</tr>
<tr>
<td>is_node</td>
<td>6</td>
</tr>
<tr>
<td>is_place</td>
<td>7</td>
</tr>
<tr>
<td>is_transition</td>
<td>7</td>
</tr>
<tr>
<td>logical</td>
<td>13</td>
</tr>
<tr>
<td>marked_petrinet, 2, 5–8, 8, 12–15</td>
<td></td>
</tr>
<tr>
<td>marking</td>
<td>8</td>
</tr>
<tr>
<td>n_flows (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>n_nodes (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>n_places</td>
<td>9</td>
</tr>
<tr>
<td>n_transitions (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>nodes</td>
<td>8</td>
</tr>
<tr>
<td>parse_trace</td>
<td>10</td>
</tr>
<tr>
<td>parse1_trace</td>
<td>10</td>
</tr>
<tr>
<td>part_of</td>
<td>11</td>
</tr>
<tr>
<td>petrinet, 2, 3, 5–8, 11, 12–15</td>
<td></td>
</tr>
<tr>
<td>petrinetR, 11</td>
<td></td>
</tr>
<tr>
<td>places</td>
<td>12</td>
</tr>
<tr>
<td>post_set</td>
<td>12</td>
</tr>
<tr>
<td>pre_set</td>
<td>13</td>
</tr>
<tr>
<td>read_PN</td>
<td>13</td>
</tr>
<tr>
<td>rename_places (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>rename_transitions (n_places)</td>
<td>9</td>
</tr>
<tr>
<td>render_PN</td>
<td>14</td>
</tr>
<tr>
<td>transitions</td>
<td>14</td>
</tr>
<tr>
<td>visNetwork_from_PN</td>
<td>15</td>
</tr>
</tbody>
</table>