

Renew Plugin FAFORMALISM – User Guide

Release 3.0
26. Januar 2026

Preamble

In version 3.0, a new formalism was added to the FAFORMALISM plugin, the FAAUTOMATONCOMPILER. The existing formalism was renamed to FANETCOMPILER. Each formalism is described in its own section. Both formalisms allow for simulating *fa*-drawings created by the FA plugin.

FANETCOMPILER

Introduction

On the following pages, you will learn about the usage of the FANETCOMPILER. This formalism interprets arcs as Java-expressions, *guards*, up- or downlinks. Other arc inscriptions have no effect on the simulation. A drawing with examples for the different kinds of arc inscriptions can be found within the *samples*-directory of the FA plugin.

The FANETCOMPILER allows the simulation of simple automaton¹ and automaton in a multi-formalism setting.

The following subsections describe the creation of a simulatable *multiformalism* system². Also, the subsequent start of a simulation thereof is covered. All explanations are given with an example of a multiformalism system which consists of models using the reference net and the finite automata formalism.

Simulations of simple automata models are not part of this user guide. A comprehensive guide on drawing automata models can be found in the user guide of the corresponding FA plugin.

Simulation setup

1. Draw the diagrams
 - Automata model(s)
 - Reference net model(s)

¹Simple automaton refers to automaton which do not accept a language, but only visualize state transitions, regardless of arc inscriptions.

²With *multiformalism system* a set of cooperating models that use more than one formalism is meant.

2. Save the diagrams

3. Select the reference net compiler

- Select the JavaNetCompiler via

```
Simulation ► Formalism
```

4. Export the reference net(s) to ShadowNetSystem

- Via

```
File ► Export ► Export current drawing ► ShadowNetSystem  
current drawing
```

5. Prepare the simulation

- Close the reference net(s)
- Choose the FANETCOMPILER via

```
Simulation ► Formalism
```

Simulation start

- Execute the RENEWcommand `startsimulation`

```
startsimulation pfaad/zum/shadowNetSystem.sns  
referenznetzname -i
```

-
- Supply the command with the path to the `.sns` file, the reference nets name and the optional `-i` option
- Option `-i` causes the simulation to be initialized; it can then subsequently be stepped through or run through in a whole step

View the simulation while running

The previously explained actions let you simulate your models without a graphical representation. To be able to view the simulation while its running, you need to do the following steps after starting the simulation as explained before.

1. Open the reference net in RENEW

2. View the simulation log

- Via hot key `Ctrl+L`
- Alternatively via

```
Simulation ► show simulation trace
```

3. View the net or automata instance

- Via right clicking on the line `New net instance ...created` and selecting `show net instance`
- Alternatively via double clicking on a reference in a diagram of the simulation

FAAUTOMATONCOMPILER

Introduction

This chapter describes the usage of the FAAUTOMATONCOMPILER. In this formalism, arc inscriptions are interpreted as letters. Arcs, therefore, can only be inscribed with single letters, which may be separated with commas. If no letters are specified, epsilon is assumed. For PDAs, the syntax is of the form $*,y->z$, where x , y and z are single letters or numbers. Multiple rules are expected to be separated by a new line. A space or the empty string ϵ is considered as an epsilon. For PDAs, the minimal inscription is therefore $,->$. Arcs without any inscriptions are interpreted as ϵ -arcs.

The simulated automaton can process words, or build them up. As of this release, the simulation of NFAs and Buechi-automatons is supported. The simulation may be automatic or manual. The next subsections will explain the effect of these different options and how they can be selected.

Syntax of Expressions

The user can add a (ω -)regular expression to a start state. This expression must be syntactically correct. A correct expression r is defined as:

- r is a single letter $[0-9a-zA-Z]$ oder the empty string ϵ .
Example: $r = a$
- r is a concatenation of regular expressions r_1 and r_2 .
Example: $r = r_1 r_2$
- r describes a Kleene star.
Example: r^* or r^*
- r describes a Kleene plus.
Example: r^+
- r describes an ω -closure.
Example: r° or r°
- r is a union over two regular expressions r_1 and r_2 .
Example: $r = r_1 + r_2$ or $r = r_1 | r_2$
- r is bracketed.
Example: (r)

Settings

Settings regarding the FAAUTOMATONCOMPILER can only be chosen, if the correct formalism is set and no simulation is running.

1. Select the FAAUTOMATONCOMPILER via

Simulation ► Formalisms ► FA Automaton Compiler

2. Open the settings menu via

A new window has opened, which allows the selection of different options and settings. Selecting an option automatically sets it, having the options is not necessary.

Simulate vs Build Word Mode

In the settings menu, one of two modes may be chosen. In the *Simulate Word* mode, the automaton processes an expression that was added by the user. Arcs may only be traversed if the expression contains a word that starts with the same letter as the arc is inscribed with. Additionally, for PDAs, the element on the top of the stack needs to be the same letter as the arc's inscription's stack instruction. By changing the state, the first letter of the expression is consumed, thereby possibly changing the expression and the stack is updated accordingly. An example of this would be $(a + b)^+.$ This expression makes it possible to traverse arcs without an inscription (ε -arcs), or arcs inscribed with a , b , or a, b . If the arc is not ε -arc, the expression will be interpreted as either $a(a + b)^*$ or $b(a + b)^*$. The first letter is consumed when traversing the arc, the resulting expression is $(a + b)^*$. If the expression contains the empty string and an end state has been reached, the state is highlighted in green to signal the acceptance of the expression by the automaton.

In the *Build Word*, every arcs can always be traversed. For PDAs, every arc can only be traversed if the stack can be updated correctly according to the inscription. The arc's inscription will be concatenated to the existing word and for PDAs the stack is updated.

Selection of the automaton

The settings menu allows the user to select which automaton model the drawing is to be interpreted as. Currently, NFAs and Buchi automata are supported.

The syntax check of the expressions takes the selected automaton type into account. If the drawing is interpreted as an NFA, the expression may not contain an ω closure. If, on the contrary, the Buchi model is chosen, every word described by the expression must end on an ω closure.

Manual simulation

In the settings menu, the user can choose between a manual or automatic simulation. Both types of simulation can be initialized via RENEW's simulation menu with *Run simulation* (Ctrl + R) or *Simulation (Net) Step* (Ctrl (+ Shift) + I).

In the manual simulation, the user has to activate and traverse arcs by selecting them with a right click, independent of RENEW's chosen simulation. If an automatic simulation is chosen and started with *Run simulation*, the arcs are traversed randomly and as long as possible. If the automatic simulation is started as *Simulation (Net) Step*, the arcs can be traversed both by right clicking them and *Ctrl + I*.

The automatic simulation is only enabled for the *Simulate Word* mode. The *Build Word* mode only allows for manual simulations, since an automatic simulation would quickly lead to a stack overflow (due to the length of the word).