`Simplx 4` release documentation

J. Krivine*

June 18, 2009

1 The command line arguments

1.1 General options

Miscellaneous

`--version` print simplx version

`--output-dir [dir]` (by default current directory) directory in which to output any produced file. Directory should already exist or Simplx raises an error.

`--compile [kappa_file]` Parses the given file an pretty print the result as well as possible compilation errors.

`--xml-session-name [name]` name for the xml file that will contain the results of the current session (default simplx.xml)

`--generate-map [kappa_file]` name of the kappa file for which the influence map should be computed.

`--no-maps` do not output inhibition/activation maps in the xml file.

`--merge-maps` will compute negative and positive influence map in the xml output.

1.2 Simulations

Generic options

`--time [time]` (by default infinite) set maximum simulation time (in rate dependent time units).

`--event [integer]` (by default infinite) set maximum number of rule applications per simulation.

*jean.krivine@plectix.com

`--rescale [real]` rescaling factor (eg. '10.0' or '0.10') –binary kinetic rates are corrected accordingly.

`--seed [integer]` seed the random generator using given integer (same integer will generate the same random number sequence)

`--max-clashes [integer]` Maximum number of null events before aborting simulation (default is 10000), use -1 for infinite.

`--memory-limit [MB]` limit the usage of the memory (in Mb). Default is infinite.

Time courses

`--sim [kappa_file]` name of the kappa file containing the kappa-rules and possibly observables, perturbations and initial conditions.

`--points [integer]` (default 1000) number of data points per plots. The points will be regularly placed on the time axis.

`--plot [name]` In addition of the xml output, this option will trigger the creation of a file containing the simulation data in space separated format together with a [file].gplot containing the gnuplot commands for displaying time course.

Causal flows

`--cflow [kappa_file]` name of the kappa file containing the kappa rules as well as at least one causal target (instruction `%causal:`).

`--dot-output` is addition of the xml output, this option will cause simplx to output causal flows as graphs in dot format (requires dot viewer like graphviz).

`--iteration [integer]`: number of attempts to obtain causal flows. Any simulation reaching successfully the observable, or reaching time or event limit (see `--event` and `--time` options) will trigger a new attempt.

`--no-compression` By default outputed causal flows are compressed. Use this option to disable compression (will increase causal simulation efficiency).

`--weak-compression` Uses a weak form of compression for causal flows (fastet than the strong one by default).

`--init [time]` Start to analyse the causal flow of the simulation at given time (in kinetic dependent time units).

`--no-arrow-closure` do not perform arrows transitive closure when displaying causal flows.

`--quotient-refinements` when the refinement of a rule r is applied during the causal analysis it will be replace r in the compuation of the flow (note that r has to exist in the kappa file).

Saving computations.

`--snapshot-at [time]` takes a snapshot current mixture at specified time unit (may use option several times). Snapshots are included in the xml file after the tag `<FinalState>`.

`--mixture-file-scheme [name]` Naming scheme for the files used to save mixtures

`--save-mixture-at [time]` Save mixture at specified time (can be used multiple times). The file is not human readable but can be loaded as initial state for a simulation using the option `--load-mixture`.

`--load-mixture [mixture_file]` Use given mixture file as initial conditions (`%init:` instructions in the kappa file will be ignored)

`--state-file-scheme [name]` Naming scheme for files used to save simulation states.

`--save-state-at [time]` Save simulation state at specified time (can be used multiple times). The file is not human readable but can be loaded for continuing the simulation using the option `--load-state`.

`--load-state [state_file]` Load given simulation state (only `%mod:` instruction will be parsed from the kappa file, the rest will be ignored).

2 The Kappa file

Below is an example of kappa file content:

```
#rules

'AB' a(x),b(x) -> a(x!1),b(x!1)  @ 0.01
'A..B' a(x!1),b(x!1) -> a(x),b(x) @ 1
'C@s2' a(x),c(x~p,y~u) -> a(x),c(y~p,x~p) @ 0.01
'C@s1' a(x!1),b(x!1),c(x~u) -> a(x!1),b(x!1),c(x~p) @ 0.01

#initial mixture

%init: 1000 * (a(x),b(x),c(x~u,y~u))

#observables and variables
```

```
%obs: 'C**' c(y~p)
%obs: a(x!_)
%obs: 'C*' c(x~p,y~u)
%obs: 'A..B'
%var: 'C' c()

#causal flow analysis

%causal: 'C@s2'
%causal: {'A..B','C@s1'} => 'C@s2'

#perturbation
%mod: (['C*'] > 2*['C']/5) & ($T>0.4) do 'A..B':= 100.0
```

2.1 Rules

The kappa file contains rules that can be *named* or *anonymous*. When one wants to plot the activity of a rule (the number of instances it has multiplied by its kinetic rate) in a time course, this rule has to be named. For instance, in the kappa file above, the instruction `%obs:'A..B'` is making the activity of the dissociation rule `'A..B'` observable (*i.e* plotted in the time course). Note also that named rule kinetic rate can be modified by perturbations (see Perturbations section below). The rules are ignored in case the option `--load-state` is used.

2.2 Initial conditions

The instruction `%init:` followed by a multiplication factor and a kappa expression allows one to declare the initial mixture to which the rules will be applied. This instruction is ignored in case the options `--load-mixture` or `--load-state` are used. Note that several `%init` lines can be used in which case mixture will be added together to form the initial state.

2.3 Time course observables and variables

Instruction `%obs:` followed by a kappa expression E declares that the number of instances of E should be plotted in the time course. The instruction `%var:` followed by a named kappa expression E simply binds the name to E for further reference in perturbations (see Perturbations section below). Note that a named observable, as:

```
%obs:'C**' c(y~p)}
```

in the above kappa file declares the kappa expression to be at the same time an observable and a variable. Variable and observable instructions are ignored if option `--load-state` is used.

2.4 Causal flow analysis

The instruction `%causal:` is used when a causal flow analysis is required (`--cflow` argument in the command line). In the above kappa file, the line:

```
%causal: {'A..B','C@s1'} => 'C@s2'
```

declares that one is interested to obtain causal flows containing rule applications of `'A..B'` and `'C@s1'`, leading to the application of rule `'C@s2'`. Note that the instruction:

```
%causal: 'C@s2'
```

is short for:

```
%causal: {} => 'C@s2'
```

that does not require any particular rule to be in the computed flow for `'C@s2'`.

2.5 Perturbations

Instruction `%mod:` allows one to define perturbations that will be applied during simulation, when given preconditions are matched. For instance, in the above kappa file, the line:

```
%mod: (['C*'] > 2*['C']/5) & ($T>0.4) do 'A..B':= 100.0
```

defines a perturbation that sets the dissociation rate of the complex `ab` to 100.0 once the number of `'C*'`, previously defined by a named observable, reaches 2/5 of the number of `'C'`s (also defined as a variable) and if the current time is greater than $0.4s$ (`$T` is the symbol for time). Note that once the preconditions are matched, the perturbation is applied and discarded. For now (version 4.0), only perturbation that change the kinetics of an existing rule may be defined. Therefore, a perturbation that adds 1000 agents, say $d(x)$, at 3 sec should be declared as:

```
'intro d'  -> d(x) @ 0.0 #introduction of d(x) at rate 0.0

%var: 'D' d(x) #binding 'D' to d(x)

%mod: $T>3 do 'intro d':=$INF  # when current time is greater \
                               than 3 sec. introduce D\
                               infinitely fast

%mod: &\tt ['D']>1000 do 'intro d':=0.0 # when 'D' reaches 1000\
                                          units, stop \
                                          introducing d(x)
```