Event Based Markovian Simulation

The $\Psi - 3$ software

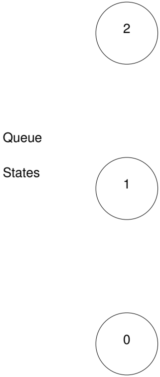
Benjamin.Briot@inria.fr Jean-Marc.Vincent@imag.fr

Laboratoire d'Informatique de Grenoble,
Inria team MESCAL
University Grenoble-Alpes, France

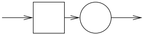
Marmote Workshop,
Montpellier 2015, May 29

Outline

Events and Poisson Systems

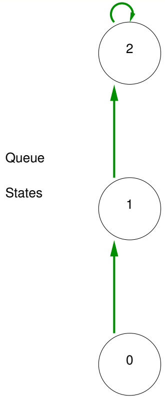


$M/M/1$ capacity $C = 2$

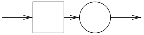


⇒ monotone events

Events and Poisson Systems

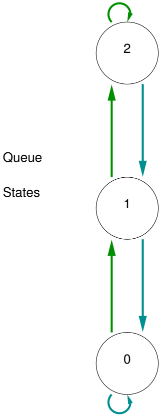


$M/M/1$ capacity $C = 2$

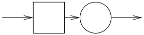


$\Rightarrow$ monotone events

Events and Poisson Systems

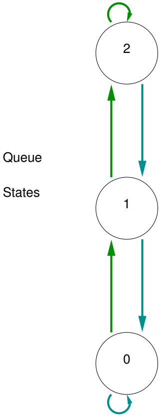


$M/M/1$ capacity $C = 2$

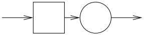


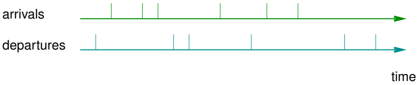
$\Rightarrow$ monotone events

Events and Poisson Systems



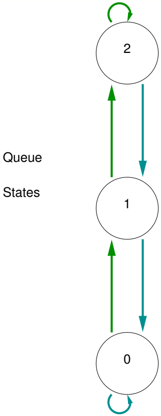
$M/M/1$ capacity $C = 2$



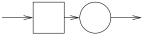


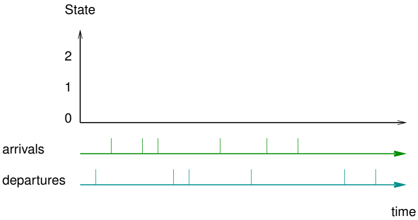
⇒ monotone events

Events and Poisson Systems



$M/M/1$ capacity $C = 2$



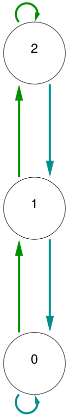


⇒ monotone events

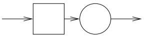
Events and Poisson Systems

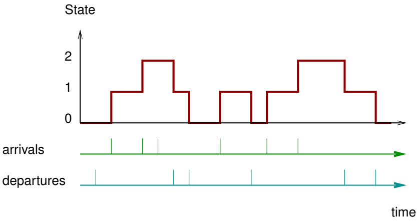
Queue

States



$M/M/1$ capacity $C = 2$



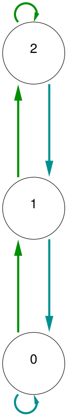


$\Rightarrow$ monotone events

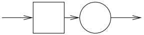
Events and Poisson Systems

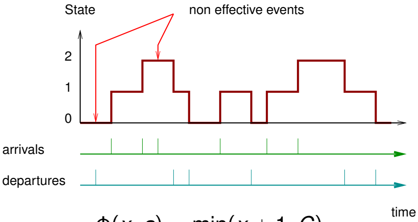
Queue

States



$M/M/1$ capacity $C = 2$

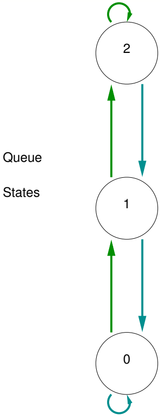




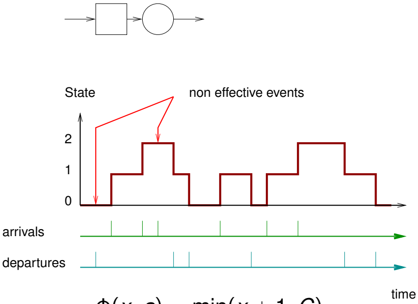
$$\Phi(x, a) = \min(x + 1, C)$$
$$\Phi(x, d) = \max(x - 1, 0)$$

$\Rightarrow$ monotone events

Events and Poisson Systems



$M/M/1$ capacity $C = 2$

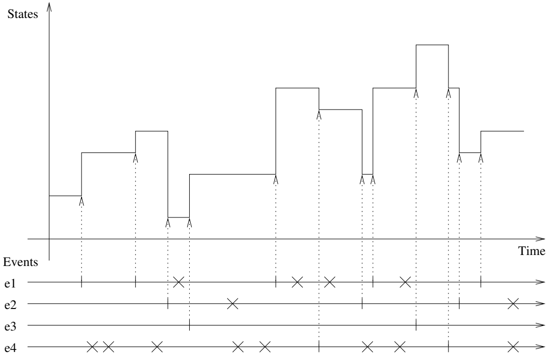


$$\Phi(x, a) = \min(x + 1, C)$$
$$\Phi(x, d) = \max(x - 1, 0)$$

$\Rightarrow$ **monotone events**

Event Modelling

Multidimensional state space : $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_K$ with $\mathcal{X}_i = \{0, \cdots, C_i\}$.
Event e :
$\rightsquigarrow$ transition function $\Phi(., e)$; (skip rule)
$\rightsquigarrow$ Poisson process λ_e



1781-1840

Event modelling : Uniformization

$$\Lambda = \sum_e \lambda_e \text{ and } \mathbb{P}(\textit{event } e) = \frac{\lambda_e}{\Lambda};$$

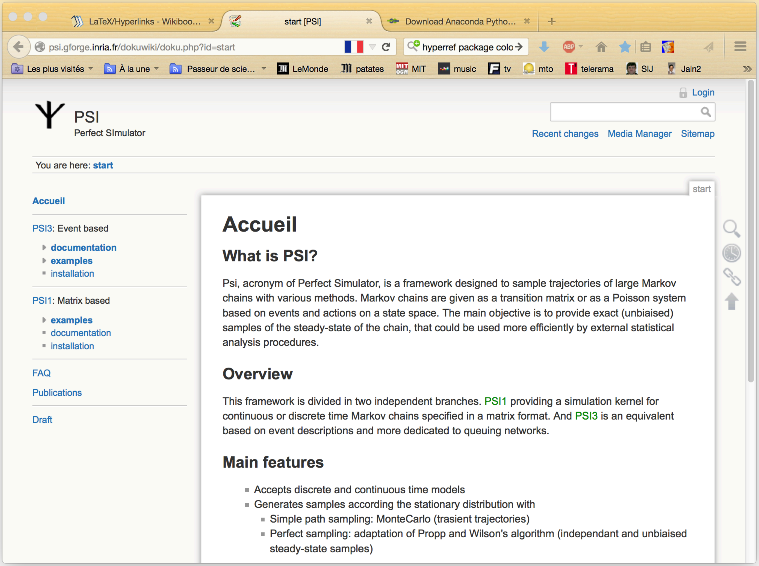
Trajectory : $\{e_n\}_{n\in\mathbb{Z}}$ i.i.d. sequence.

⇒ Homogeneous Discrete Time Markov Chain [Bremaud 99]

$$X_{n+1} = \Phi(X_n, e_{n+1}).$$

Generation among a small finite space $\mathcal{E}$: $\mathcal{O}(1)$

Ψ software



Accueil

What is PSI?

Psi, acronym of Perfect Simulator, is a framework designed to sample trajectories of large Markov chains with various methods. Markov chains are given as a transition matrix or as a Poisson system based on events and actions on a state space. The main objective is to provide exact (unbiaised) samples of the steady-state of the chain, that could be used more efficiently by external statistical analysis procedures.

Overview

This framework is divided in two independent branches. PSI1 providing a simulation kernel for continuous or discrete time Markov chains specified in a matrix format. And PSI3 is an equivalent based on event descriptions and more dedicated to queuing networks.

Main features

- Accepts discrete and continuous time models
- Generates samples according the stationary distribution with
  - Simple path sampling: MonteCarlo (trasient trajectories)
  - Perfect sampling: adaptation of Propp and Wilson's algorithm (independant and unbiaised steady-state samples)

Software architecture

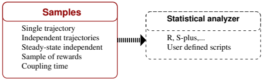
Aim of the software

- finite capacity queueing network simulator
- rare events estimation (rejection, blocking,...)
- statistical guarantees (independence of samples)

$\Rightarrow$ **Simulation kernel**

- open source (C, GPL licence)
- extensible library of events
- multiplatforms (linux (debian), mac OSX,...)

Workflow


Samples
Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time
Statistical analyzer
R, S-plus,...
User defined scripts

Workflow

Model libraries

- **Queues description**
  - servercs, capacities
- **Event description**
  - Rate
  - Activation condition
  - Action of the event

Simulation kernels

- **Forward sampling**
  - trajectories
- **Backward sampling**
  - Monotone
  - Envelopes
  - Envelopes and split

Samples

- Single trajectory
- Independent trajectories
- Steady-state independent
- Sample of rewards
- Coupling time

→

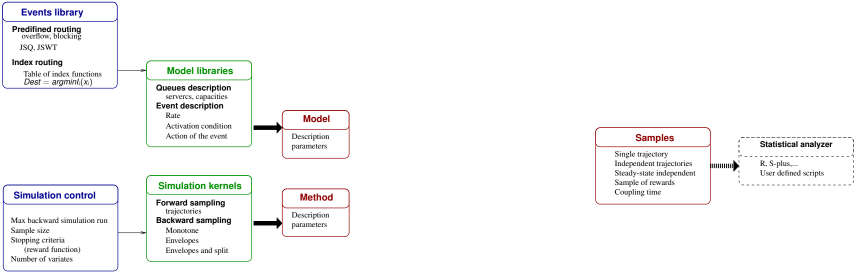
Statistical analyzer

- R, S-plus,...
- User defined scripts

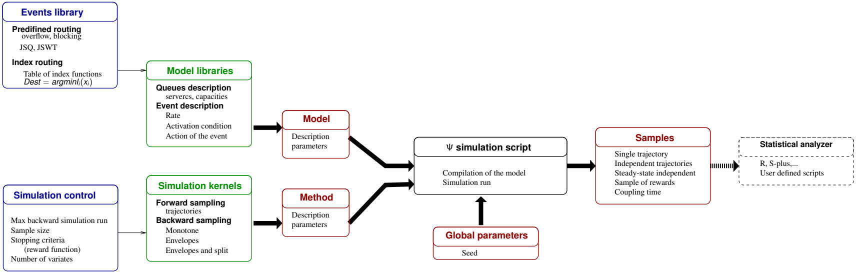
Workflow


Events library
Predifined routing
overflow, blocking
JSQ, JSWT
Index routing
Table of index functions
$Dest = argminI_i(x_i)$
Model libraries
Queues description
servercs, capacities
Event description
Rate
Activation condition
Action of the event
Simulation control
Max backward simulation run
Sample size
Stopping criteria
(reward function)
Number of variates
Simulation kernels
Forward sampling
trajectories
Backward sampling
Monotone
Envelopes
Envelopes and split
Samples
Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time
Statistical analyzer
R, S-plus,...
User defined scripts

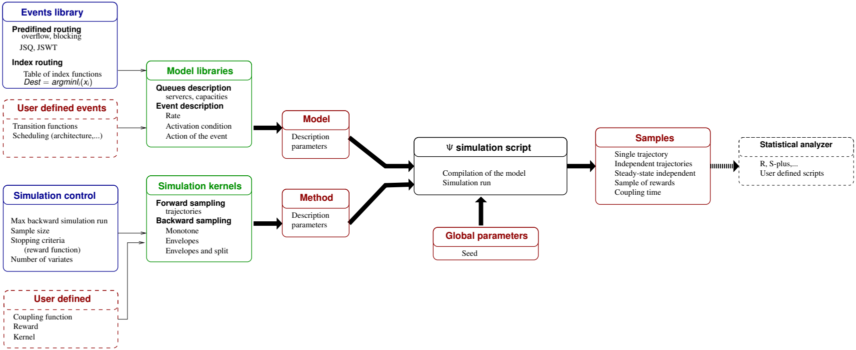
Workflow


Events library
Predifined routing
overflow, blocking
JSQ, JSWT
Index routing
Table of index functions
$Dest = argminI_i(x_i)$
Model libraries
Queues description
servercs, capacities
Event description
Rate
Activation condition
Action of the event
Model
Description
parameters
Simulation control
Max backward simulation run
Sample size
Stopping criteria
(reward function)
Number of variates
Simulation kernels
Forward sampling
trajectories
Backward sampling
Monotone
Envelopes
Envelopes and split
Method
Description
parameters
Samples
Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time
Statistical analyzer
R, S-plus,...
User defined scripts

Workflow


Events library
Predifined routing
overflow, blocking
JSQ, JSWT
Index routing
Table of index functions
$Dest = argmin I_i(x_i)$
Model libraries
Queues description
servercs, capacities
Event description
Rate
Activation condition
Action of the event
Model
Description
parameters
Simulation control
Max backward simulation run
Sample size
Stopping criteria
(reward function)
Number of variates
Simulation kernels
Forward sampling
trajectories
Backward sampling
Monotone
Envelopes
Envelopes and split
Method
Description
parameters
Ψ simulation script
Compilation of the model
Simulation run
Global parameters
Seed
Samples
Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time
Statistical analyzer
R, S-plus,...
User defined scripts

Workflow


Events library
Predifined routing
overflow, blocking
JSQ, JSWT
Index routing
Table of index functions
$Dest = argmin I_i(x_i)$
User defined events
Transition functions
Scheduling (architecture,...)
Model libraries
Queues description
servercs, capacities
Event description
Rate
Activation condition
Action of the event
Model
Description
parameters
Ψ simulation script
Compilation of the model
Simulation run
Samples
Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time
Statistical analyzer
R, S-plus,...
User defined scripts
Simulation control
Max backward simulation run
Sample size
Stopping criteria
(reward function)
Number of variates
Simulation kernels
Forward sampling
trajectories
Backward sampling
Monotone
Envelopes
Envelopes and split
Method
Description
parameters
Global parameters
Seed
User defined
Coupling function
Reward
Kernel

Modeling

Syntax

```
Queues:
    - id:              [id_value]
    [parameter_1]:     [parameter_2_value]
    ...
```

Example

```
Queues:
    - id:               queue1
      min:              0
      max:              50

    - id:               queue2
      min:              0
      max:              80
```

Modeling

Short list

- **ext_arrival_reject** One client comes from outside to the listed queues with priorities (the list order). The client is rejected if not possible.
- **ext_departurer** One client lives the specified queue.
- **routing_n_queues_reject** Select a client from the listed queues and route to another queue. Client is rejected if routing is not possible.
- **routage_nfile_bloc** Select a client from the listed queues and route to another queue. Client stays on the origin queue if routing is not possible.
- **JSQ_rejet** Select a client from the listed queues and route to another queue with the least clients. Client is rejected if routing is not possible.

Modeling

Syntax

```
Events:
    - id:             [id_value]
    [parameter_1]:    [parameter_2_value]
    ...
```

Example

```
Events:
    - id:               evt1
      type:             Default$ext_arrival_reject
      rate:             1.6
      from:             [outside]
      to:               [queue1,  drop]

    - id:               evt2
      type:             Default$ext_departure
      rate:             2.0
      from:             [queue1]
      to:               [drop]
```

Simulation

Single trajectory sampling (Monte Carlo)

- Simple forward
- Simple forward parallel

Perfect sampling (Propp & Wilson) and extensions

- Bakward Monotone
- Bakward Envelope
- Bakward Envelope Splitting

Outline

Installation

- Unix-like operating system: GNU/Linux, MacOSX
- C/C++ compiler
  - GCC >= 4.4 (for OpenMP 3.0 support)
- CMake > 3.0

Installation

PSI3 will be installed in `/usr/local`.

In the source directory:

```
cmake .
make
sudo make install
```

Installation

Installation destination can be changed through `CMAKE_INSTALL_PREFIX`. For instance, installing in `$HOME/psi3`:

In the source directory:

```
DESTDIR="$HOME/psi3"
cmake -D CMAKE_INSTALL_PREFIX="$DESTDIR" .
make
sudo make install
```

Then, you can update your Path to run PSI3:

In shell:

```
export PATH="$DESTDIR/bin:$PATH"
```

Installation

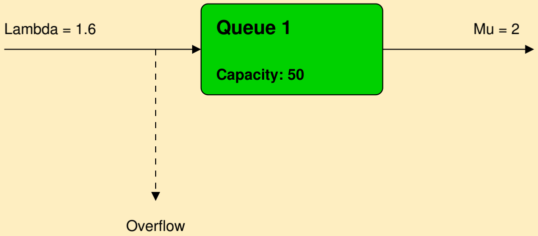
Notes for mac user

- Although PSI3 can compile and works with clang, our advise is to use gcc.
- If you don't have gcc, you can install it through homebrew (http://brew.sh/)
- If notice CMAKE still use clang. Check the value of `CMAKE_C_COMPILER`. CCMAKE offers a graphical front-end that is often a good choice to get a good overview of variables used by CMAKE.

Outline

Example

Overview

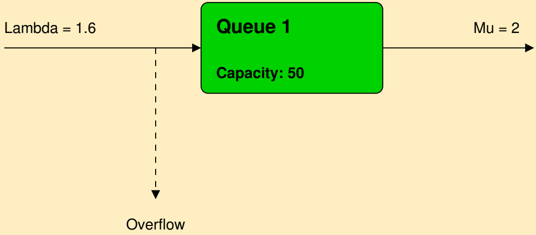


1 Queue

```
- id:          queue1
  min:         0
  max:         50
```

Example

Overview

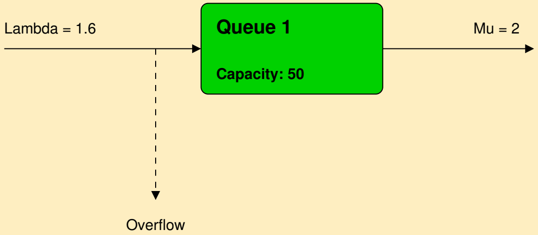


2 Events

- ext_arrival_reject
- ext_departure

Example

Overview

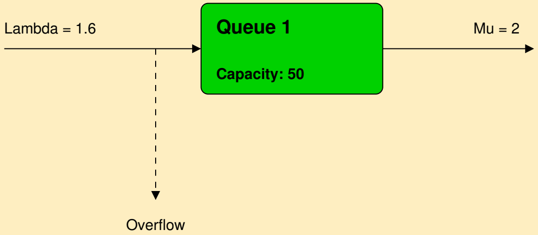


Event 1

```
- id:       evt1
  type:     Default$ext_arrival_reject
  rate:     1.6
  from:     [outside]
  to:       [queue1, drop]
```

Example

Overview



Event 2

```
- id:              evt2
  type:            Default$ext_departure
  rate:            2.0
  from:            [queue1]
  to:              [drop]
```

Example

model.yaml


```
Queues:
    - id:                 queue1
      min:                0
      max:                50
Events:
    - id:                 evt1
      type:               Default$ext_arrival_reject
      rate:               1.6
      from:               [outside]
      to:                 [queue1,  drop]
    - id:                 evt2
      type:               Default$ext_departure
      rate:               2.0
      from:               [queue1]
      to:                 [drop]
```

Example

Simpleforward

To generate a trajectory of this model.

simpleforward.yaml

```
Method:                   simpleforward

# Sample will have one million states
TrajectoryLength:         1000000

# Initial states of queues.
# "~" means to let PSI3 choose it randomly.
InitialState :            ~
```

Example

param.yaml


```
# Random generator seed. "~" for random seed
Seed:                   7

# Configuration of model
PrintModel:             Yes

# Parameters of simulation
PrintParam:             Yes

# Total time of the simulation from begin to end
PrintSimulationTime:    Yes
```

Example

Execution

```
> psi3_unix -m simple-queue.yaml -p param.yaml
-k simpleforward.yaml

Begin user files compilation...
OK
Begin simulation...
Number total of transition calls: 1000000

# Total simulation time  : 297.071000 milli-seconds
End of simulation
```

Example

output.txt (troncated)

```
# PSI3 INFO:
# version: 1.3.0
# build type: DEBUG
# C compiler: GNU
# compiler options: -Werror -Wall -rdynamic

# General Param:
#  seed: 7
# Method:
#  TrajectoryLength: 1000
#  InitialState:  24

...
```

Example

output.txt (truncated)

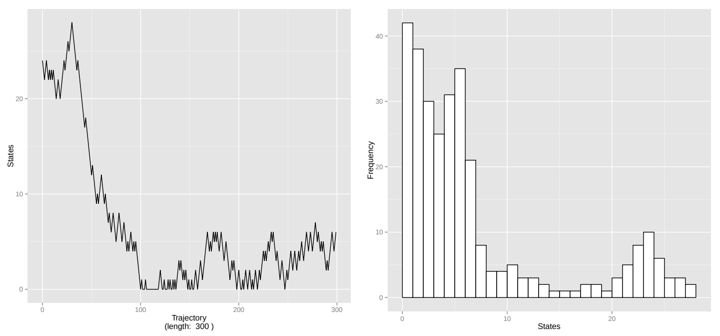
```
# Output data:
0  Na 24
1  1  23
2  1  22
3  0  23
4  0  24
5  1  23
...
999998   1  0
999999   0  1

# Number total of transitions calls: 999999
# Total simulation time: 233.257000 milli-seconds
```

Example

With a simple script R available on this page:
http://psi.gforge.inria.fr/dokuwiki/doku.php?id=psi3:examples

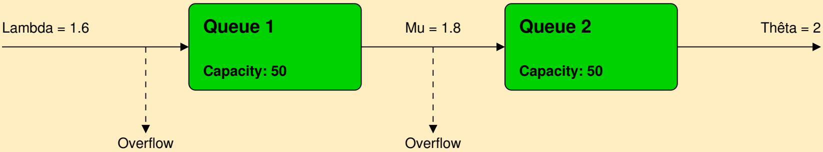
R output



Outline

Example

Overview



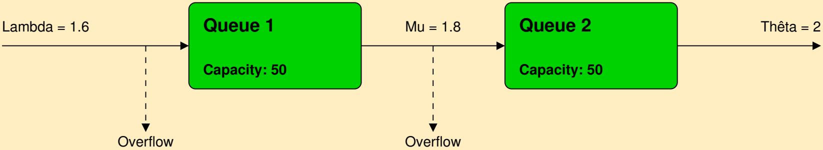
2 Queues

```
Queues:
    - id:            queue1
      min:           0
      max:           50

    - id:            queue2
      min:           0
      max:           50
```

Example

Overview

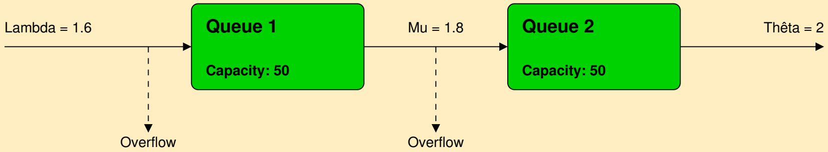


2 Events

- ext_arrival_reject
- routing_n_queues_reject
- ext_departure

Example

Overview

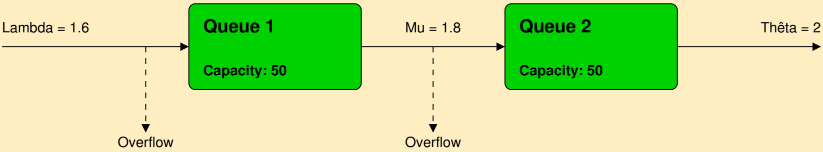


Event 1

```
- id:          evt1
  type:        Default$ext_arrival_reject
  rate:        1.6
  from:        [outside]
  to:          [queue1, drop]
```

Example

Overview

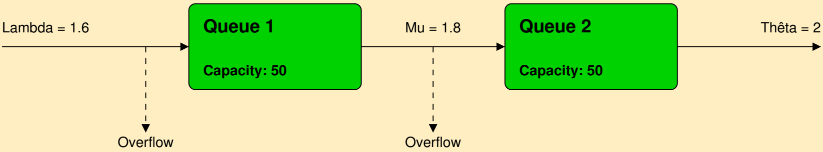


Event 2

```
- id:           evt2
  type:         Default$routing_n_queues_reject
  rate:         1.8
  from:         [queue1]
  to:           [queue2]
```

Example

Overview



Event 3

```
- id:           evt2
  type:         Default$ext_departure
  rate:         2.0
  from:         [queue1]
  to:           [drop]
```

Example

BackwardMonotone

To sample 1000 steady states of the system..

method.yaml


```
# Simulation algorithm
Method:            backwardmonotone

# Sample number
SampleNumber:      1000

# Number of Antithetic variable
Antithetic:        1

# Doubling period (Yes or No)
Doubling:          Yes

# Size of maximal trajectory
TrajectoryMax:     3000000
```

Example

param.yaml

```
# Random generator seed. "~" for random seed
Seed:                   7

# Configuration of model
PrintModel:             Yes

# Parameters of simulation
PrintParam:             Yes

# Total time of the simulation from begin to end
PrintSimulationTime:    Yes
```

Example

Execution

```
> psi3_unix -m simple-queue.yaml -p param.yaml
-k simpleforward.yaml

Begin user files compilation...
OK
Begin simulation...
InitMemoryLength: automatic default value (1000)

# Total simulation time: 287.465000 milli-seconds
End of simulation
```

Example

output.txt (truncated)

```
# PSI3 INFO:
# version: 1.3.0
# build type:
# C compiler: GNU
# compiler options: -Werror -Wall -rdynamic

# General Param:
#  seed: 3
# Method:
#  SampleNumber: 1000
#  TrajectoryMax: 3000000
#  Antithetic: 1
#  Doubling: 1
#  InitMemoryLength: 1000
...
```

Example

output.txt (truncated)

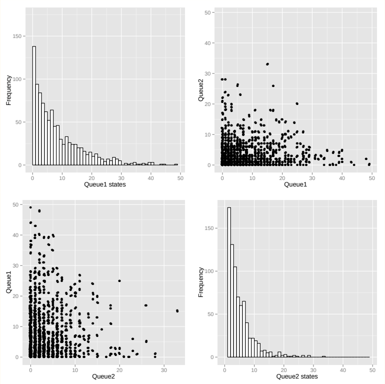
```
# Output data:
0   14  7   - 10
1   2  8   - 11
2   2  3   - 11
3   3  4   - 11
...
996    0  3   - 10
997    3  1   - 10
998    6  0   - 10
999    15  1   - 10

# Total simulation time: 287.465000 milli-seconds
```

Example

With a simple script R available on this page:
http://psi.gforge.inria.fr/dokuwiki/doku.php?id=psi3:examples

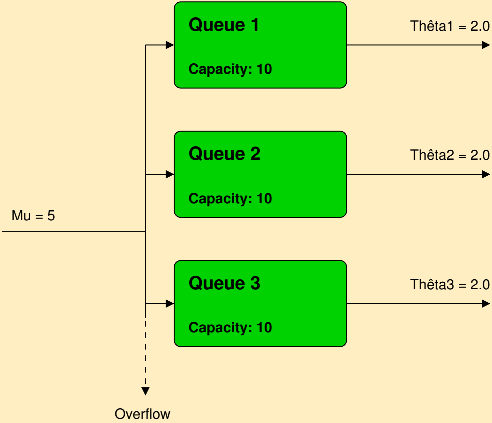
R output



Outline

Example

Overview



Two kind of event

- JSQ_rejet
- ext_departure

Example

model.yaml (1)


```yaml
Queues:
    - id:               queue0
      min:              0
      max:              10
    - id:               queue1
      min:              0
      max:              10
    - id:               queue2
      min:              0
      max:              10
Events:
    - id:               evt1
      type:             Default$JSQ_rejet
      rate:             5.0
      from:             [outside]
      to:               [queue0,queue1,queue2,drop]
```

Example

model.yaml (2)


```yaml
  - id:                  evt2
    type:                Default$ext_departure
    rate:                2.0
    from:                [queue0]
    to:                  [drop]
  - id:                  evt3
    type:                Default$ext_departure
    rate:                2.0
    from:                [queue1]
    to:                  [drop]
  - id:                  evt4
    type:                Default$ext_departure
    rate:                2.0
    from:                [queue2]
    to:                  [drop]
```

Example

BackwardMonotone

- To sample 1000 steady states of the system
- User defined output function to print max and min of queues

method.yaml

```
# Simulation algorithm
Method:               backwardmonotone

# Sample number
SampleNumber:         1000

# Doubling period (Yes or No)
Doubling:             Yes

# Size of maximal trajectory
TrajectoryMax:        3000000

MyLib:               [lib/outputfct]
OutputFct:           MyLib$output_minmax
```

Example

```
void output_minmax (FILE * f, int **state,
                           int sample, int *log2stop_time)
{
   int i, n,max = 0, min = 0;

   fprintf (f, "%d \t", sample);

   for (n = 0; n < nb_AV; n++) {

      min = state[n][0];
      max = state[n][0];

      for (i = 1; i < nb_queues; i++) {
         max = max2 (max, state[n][i]);
         min = min2 (min, state[n][i]);
      }

      fprintf (f, "%d\t%d\n", min, max);
   }
}
```

Example

param.yaml

```
# Random generator seed. "~" for random seed
Seed:                    7

# Configuration of model
PrintModel:              Yes

# Parameters of simulation
PrintParam:              Yes

# Total time of the simulation from begin to end
PrintSimulationTime:     Yes
```

Example

Execution

```
> psi3_unix -m simple-queue.yaml -p param.yaml
-k simpleforward.yaml

Begin user files compilation...
OK
Begin simulation...
InitMemoryLength: automatic default value (1000)

# Total simulation time: 73.659000 milli-seconds
End of simulation
```

Example

output.txt (truncated)

```
# Output data:
0  2  3
1  3  4
2  1  2
3  2  3
4  1  2
5  2  4
...
995   2  2
996   3  4
997   0  0
998   1  3
999   0  2

# Total simulation time: 73.659000 milli-seconds
```

Example

With a simple script R

R output