The PEPA Eclipse Plug-in

Mirco Tribastone
mtribast@inf.ed.ac.uk
http://homepages.inf.ed.ac.uk/mtribast/

4 November 2008

Outline

- Overview of PEPA
- Plug-in demonstration
  - Download and installation
  - Editing a PEPA description
  - State space navigation
  - Markov chain analysis
  - Experimentation

Overview of PEPA

PEPA is a formal language for performance evaluation.
A model is described as a cooperation between sequential components.

Overview of PEPA

PEPA is a formal language for performance evaluation.
A model is described as a cooperation between sequential components.
A sequential component cycles through a set of local states.
We use the operator prefix to associate an action type and a rate with a transition:

$$(\alpha, r).P$$

Overview of PEPA

PEPA is a formal language for performance evaluation.
A model is described as a cooperation between sequential components.
A sequential component cycles through a set of local states.
We use the operator prefix to associate an action type and a rate with a transition:

$$(\alpha, r).P$$

A sequential component may enable two of more activities simultaneously. We use the operator choice for this:

$$(\alpha, r).P_1 + (\beta, s).P_2$$

Overview of PEPA

PEPA is a formal language for performance evaluation.
A model is described as a cooperation between sequential components.
A sequential component cycles through a set of local states.
We use the operator prefix to associate an action type and a rate with a transition:

$$(\alpha, r).P$$

A sequential component may enable two of more activities simultaneously. We use the operator choice for this:

$$(\alpha, r).P_1 + (\beta, s).P_2$$

We may use a constant for naming purposes

$$A \stackrel{def}{=} (\alpha, r).P$$

Example

We wish to model the following editing workflow:

- Download a document
- Edit
- Save

Example

We wish to model the following editing workflow:

- Download a document
- Edit
- Save

We may model this through a three-state sequential component:

$$
\begin{aligned}
User_1 &\stackrel{def}{=} (download, r_1).User_2 \\
User_2 &\stackrel{def}{=} (edit, r_2).User_3 \\
User_3 &\stackrel{def}{=} (save, r_3).User_1
\end{aligned}
$$

Compositional Modelling

A model for a file server:

$$\begin{aligned} Server_1 &\stackrel{def}{=} (download, s_1).Server_2 \\ Server_2 &\stackrel{def}{=} (reset, s_2).Server_1 \end{aligned}$$

Compositional Modelling

A model for a file server:

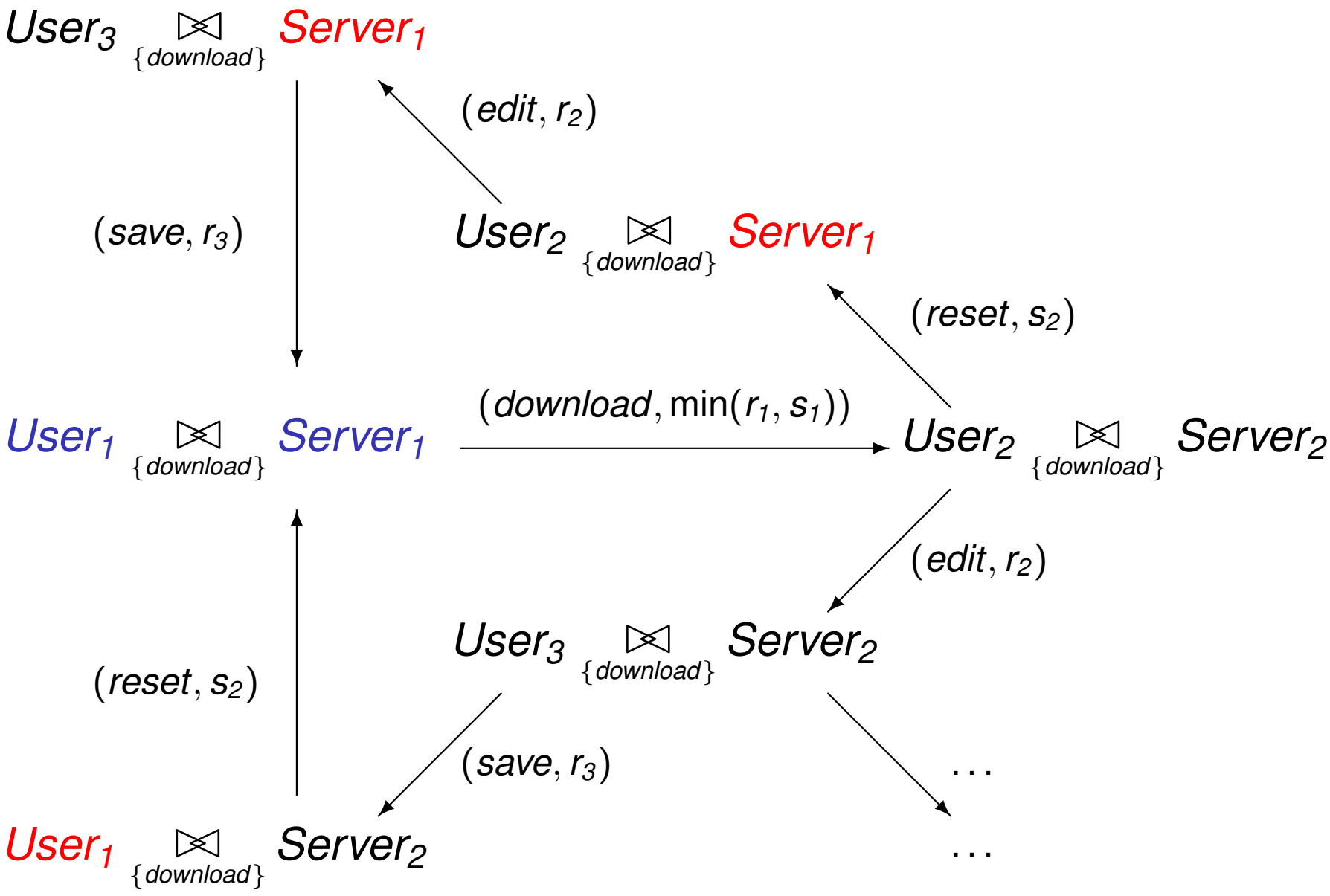
$$\begin{aligned} Server_1 &\stackrel{def}{=} (download, s_1).Server_2 \\ Server_2 &\stackrel{def}{=} (reset, s_2).Server_1 \end{aligned}$$

Consider now a server and a user together:

$$\begin{aligned} User_1 &\stackrel{def}{=} (download, r_1).User_2 \\ User_2 &\stackrel{def}{=} (edit, r_2).User_3 \\ User_3 &\stackrel{def}{=} (save, r_3).User_1 \end{aligned}$$

$$System \stackrel{def}{=} User_1 \underset{\{download\}}{\bowtie} Server_1$$

The Transition System



Demo