Introduction to the PiVizTool

Anja Bog
anja.bog@hpi.uni-potsdam.de

November 22, 2006

The PiVizTool is a tool for the simulation and analysis of mobile concurrent systems described in the π-calculus. PiVizTool has been developed by Anja Bog in the course of a master's thesis.

The tool graphically displays the linking structure of a defined π-calculus system. For the graphical representation an extended version of Robin Milner's flow graphs [3] is utilized. The evolution of the displayed π-calculus system can be influenced step-by-step through a user's interaction with the graphical representation. Thereby, the interaction behavior and link passing mobility of the π-calculus agents in the system can be monitored. Due to the visualization of the interacting π-calculus agents and their links among each other, an easier understanding of π-calculus system evolution is achieved.

The π-calculus systems as input for the tool can be described by using the same input syntax as the Mobility Workbench (MWB) [4] and Another Bisimulation Checker (ABC) [1].

1 Running PiVizTool

As a prerequisite for running the PiVizTool, an installation of the Graphviz graph visualization software [2] is required. Graphviz provides the dot graph drawing engine, which is used by the PiVizTool to automatically create the layout of the presented π-calculus system. Executable installation packages can be downloaded at:

```
http://www.graphviz.org/
```

The dot installation path can be configured in the "File → Set dot execution path..." menu.

The PiVizTool application is written in Java. It is distributed as a Java Archive (JAR) and can be run using the Java interpreter. The command for starting the PiVizTool application is:

```
java -jar PiVizTool.jar
```

2 Example

As a short introductory example to how the PiVizTool works, we imagine a situation from every-day life. Let's assume two persons: John and Mary. Both have a telephone. John wants to call Mary, but does not know her telephone

number. Luckily, Mary has registered with the directory assistance, so John can just call the directory assistance by using its publicly known number and ask for Mary's number. Now that John knows Mary's number, he is able to call her and talk to her. In this example, after John has queried Mary's telephone number, a new link that can be used for communication between John and Mary has been established. The two states of the linking structure of the system are depicted in Figures 1(a) and 1(b).

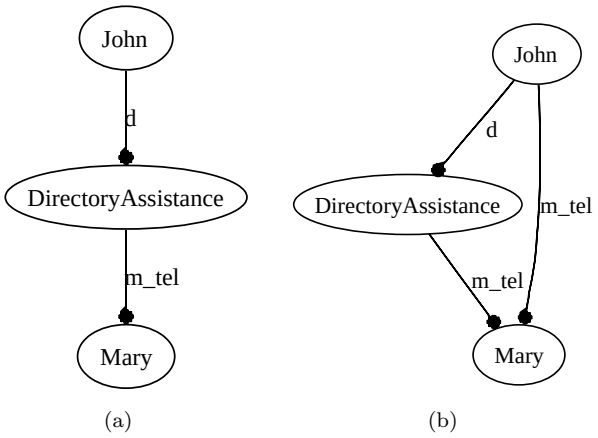

Figure 1: John and Mary

For a translation of this situation into a π-calculus system, we need to define three π-calculus agents, one for each: Mary, John and the directory assistance. Furthermore, we need the telephone numbers as names for communication:

- John's number: j_tel
- Mary's number: m_tel
- The name under which Mary has been registered with the directory assistance: $mary$
- Directory assistance's number: d

The π-calculus agent representing John is structured as follows. The publicly known name d is used to query the directory assistance for Mary's communication link. With this query, the name $mary$ is sent as an identifier, so the directory assistance knows, whose data John asks for. Additionally, John transmits his communication link, so the directory assistance can send Mary's communication data to him. Then he is able to use the received name for communication with Mary. Regarding the structure of Mary's process, she only waits until somebody sends her a message.

$$\begin{aligned} John(d, mary) &= (\mathbf{v} j\_tel)\overline{d}\langle mary, j\_tel\rangle.j\_tel(marys\_number). \\ &\quad (John(d, mary) \mid \\ &\quad (\mathbf{v} hello)\overline{marys\_number}\langle hello\rangle.\mathbf{0}) \\ Mary(m\_tel) &= m\_tel(message).Mary(m\_tel) \end{aligned}$$

To simplify the process definition for the directory assistance, we assume the state that Mary is already registered and only take into account the part that is needed for providing John with Mary's number.

$$
\begin{aligned}
&DirectoryAssistance(d, m\_tel, mary) \\
&\quad = \quad d(name, chan).(\underline{DirectoryAssistance}(d, m\_tel, mary) \mid \\
&\qquad [name = mary]\overline{chan}\langle m\_tel\rangle.\mathbf{0}) \\
&\qquad + \overline{m\_tel}\langle message\rangle.DirectoryAssistance(d, m\_tel, mary)
\end{aligned}
$$

Finally, to provide the independent agents Mary, John and Directory Assistance with a common scope for the used names, we define an agent S for the entire system:

$$
\begin{aligned}
S \quad = \quad & (\mathbf{v}m\_tel, mary, d)(John(d, mary) \mid Mary(m\_tel) \mid \\
& DirectoryAssistance(d, m\_tel, mary))
\end{aligned}
$$

To be used as input for the PiVizTool, the definitions are refined into the following ASCII representation. As can be seen, an additional tag `exec` is inserted. This tag tells the PiVizTool, which of the defined agents are to be run initially.

```
exec agent S = (^m_tel,mary,d)(John(d,mary) | Mary(m_tel) |
               DirectoryAssistance(d,m_tel,mary))

agent John(d,mary) = (^j_tel)'d<mary,j_tel>.j_tel(marys_number).
                     (John(d,mary) |
                     (^hello)'marys_number<hello>.0)

agent Mary(m_tel) = m_tel(message).Mary(m_tel)

agent DirectoryAssistance(d,m_tel,mary)
           = d(name,chan).(DirectoryAssistance(d,m_tel,mary) |
             [name = mary]'chan<m_tel>.0)
             + 'm_tel<message>.DirectoryAssistance(d,m_tel,mary)
```

Some screenshots of the PiVizTool during the simulation of this example are shown below. Before explaining the shown process some general comments regarding the user interface will be given. The user interface is separated into two parts. The left hand side gives a list of restricted names in the displayed system. From this list one or more names can be selected. Upon selection, a name's scope will be immediately shown in the graphical representation. The scope of a name is represented by connecting the name and all the nodes representing agents in the scope of the name with dashed lines. The right hand side shows the graphical representation of the current linking state of the system. Direct interaction with this graphical representation causes the π-calculus system to change states. Agents are represented by oval nodes and the links between the agents represent message sending and receiving. The receiving side is annotated by a small dot. The label of the communication link gives the information, which name is used for sending/receiving. Communications have to be distinguished into blocked and active ones. Active ones art those, whose

prefixes are not blocked by other prefixes. They are displayed as black links. Blocked communications are displayed as shaded links. Tau actions of agents that are currently executable are displayed by shading the node of the agent they belong to.

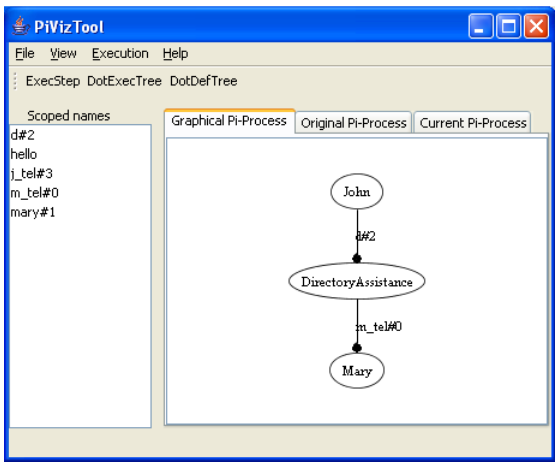

Figure 2: Screenshot: State after loading definitions into the tool

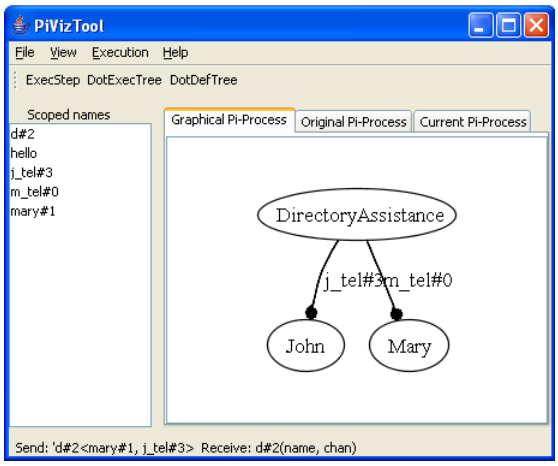

Figure 3: Screenshot: John requestet Mary's number

Figure 2 depicts the initial linking structure after the definitions have been loaded into the tool. Figure 3 shows that John has requested Mary's information from the directory assistance and is now able to receive the information via the channel `j_tel`, he has provided the directory assistance with. In Figure 4 the state of John being able to communicate with Mary by using the received link is depicted.

This concludes the short introduction to the PiVizTool. More information regarding implementation details and used data structures, as well as more sophisticated examples, can be found in the associated master's thesis "A Visual

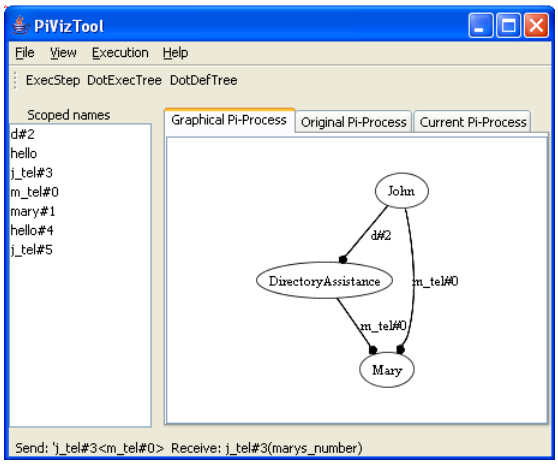

Figure 4: Screenshot: John establishing a connection with Mary

Environment for the Simulation of Business Processes based on the Pi-Calculus" by Anja Bog.

References

[1] Sebastien Briais. The ABCs User Guide. Available at: http://lamp.epfl.ch/sbriais/abc/abc.html, 2003.

[2] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. In *Software – Practice and Experience*, 30(11), pages 1203 – 1233, 2000.

[3] Robin Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, New York, NY, USA, 1999.

[4] Björn Victor and Faron Moller. The Mobility Workbench A Tool for the Pi-Calculus. In David Dill, editor, *CAV94: Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.