Graphical Languages for Functional Reactive Modeling based on Petri nets

David Mosteller, Michael Haustermann, Leonie Dreschler-Fischer
Outline

• Related works: functional approaches to modeling
• The Reference Net formalism
• Functional reactive modeling with Reference Nets
• Functional components
• RMT approach: developing domain-specific modeling tools
• Home automation example
Related Works

• Functional block diagrams
  Modelica (Tiller, 2001)
  Matlab Simulink (Xue and Chen, 2013)

• Graphical functional reactive programming
  Akka streams, Apache Spark (Davis, 2019)

• Metamodelling environments with execution semantics
  GEMOC Studio (Combemale et al., 2017)

• Functional languages and Petri nets
  Coloured Petri nets (Jensen and Kristensen, 2009)
  Curry-Coloured Petri nets (Simon et al., 2019)
Goals / Approach

Goal: unify the following characteristics (to a unique approach)

- Graphical languages → easy to follow
- Meta modeling → quick results for building tools
- Operational semantics → can execute in simulation environment
- Functional decomposition → clear notion of components, referential transparency
- Reactive programming → communicate with environment

Approach

- Develop Reference net components that capture functional properties for the specification of domain specific modeling languages using operational semantics
- Higher order functions: nets-within-nets paradigm of Reference Nets
- Model driven: generate modeling tools
Reference Nets

Tuples

Lists

Synchronous Channels

Figure 1: Using collections in Reference Nets

Figure 2: Synchronous channels of Reference Nets

Figure 3: Dynamic hierarchies: nets-within-nets [19]
Functional Reactive Systems

Figure 4: A functional perspective on Reference Nets

Figure 5: An immutable named key-value pair data structure

Figure 6: A reactive system

Figure 7: Higher-order functions
# Semantic Components

<table>
<thead>
<tr>
<th>XOR-Split</th>
<th>XOR-Merge</th>
<th>Parallel Split</th>
<th>Parallel Join</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td><img src="image5.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Start</th>
<th>Stop</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image6.png" alt="Diagram" /></td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Table 1: Mapping of graphical and Petri net constructs

<table>
<thead>
<tr>
<th>Start</th>
<th>Stop</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image9.png" alt="Diagram" /></td>
<td><img src="image10.png" alt="Diagram" /></td>
<td><img src="image11.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

With these models, a tool is generated as a Renew plug-in, as shown at the bottom right side of Figure 8. A complete example of the Rmt models and additional information about the tools can be found in our article on the Rmt framework [13].

Table 1 shows the semantic mapping, applying graphical components to semantic Reference Net components. In combination with the previously described steps, this is all it needs in order to build models that are ready to be executed within the Renew simulation environment. With the integrated sim...
Heating Control DSML

Together with the semantic mapping we developed in the previous section, the home automation DSML models can be executed within the simulation environment. Using the net generator from the RMT framework the Reference Net depicted in Figure 10 was generated from the home automation model in Figure 9. It was extended with some inscriptions to become executable within the Renew simulation environment. The state of the home automation system becomes interactively inspectable and with the adaption to an appropriate home automation API the model could be used to control a home automation system.

Related Works

Graphical modeling techniques that apply a functional decomposition have a long tradition in computer science. The basis of functional block diagrams and its many variants are the interpretation of elements that are commonly depicted by rectangles as black boxes that transform inputs to outputs (cf. structured analysis and design technique [16]). Standard languages exist to simulate mature instances of such diagrams, for instance using Modelica [18] or Matlab Simulink [21], and even approaches to verification [22]. However, they are usually self-contained and do not account for the environment as in functional reactive programming.
Figure 9: An example of a heating control modeled with a DSML for home automation. In between office hours (timestamp) the heating system is shutdown for saving energy.

Together with the semantic mapping we developed in the previous section, the home automation DSML models can be executed within the Renew simulation environment. Using the net generator from the RMT framework the Reference Net depicted in Figure 10 was generated from the home automation model in Figure 9. It was extended with some inscriptions to become executable within the Renew simulation environment. The state of the home automation system becomes interactively inspectable and with the adaption to an appropriate home automation API the model could be used to control a home automation system.

Related Works

Graphical modeling techniques that apply a functional decomposition have a long tradition in computer science. The basis of functional block diagrams and its many variants are the interpretation of elements that are commonly depicted by rectangles as black boxes that transform inputs to outputs (cf. structured analysis and design technique [16]). Standard languages exist to simulate mature instances of such diagrams, for instance using Modelica [18] or Matlab Simulink [21], and even approaches to verification [22]. However, they are usually self-contained and do not account for the environment as in functional reactive programming.
There are many programming libraries that support functional reactive programming and some of them even provide a graphical notation, like the GraphDSL using Akka streams [4]. As an advancement of the map-reduce approach to big data processing, the Apache Spark project gains support for graphical flow-based programming [12]. They also target IoT and cyber-physical systems, however, the approach taken with this contribution is somewhat different as it focuses more on the operational semantics than on the processing of huge amounts of data.

Several frameworks exist that support developers in creating their own (domain specific) modeling languages based on metamodels (ADOxx [5]) and some of them even provide means for simulation or interactive execution (GEMOC Studio [3]). The execution semantics is usually coded within the modeling environment, while with the Rmt approach it is provided by transformations to Petri nets. The Viatra eclipse project provides an event driven reactive framework for model transformations [20], which is an interesting approach for Petri net transformations.

Petri nets in general already offer a functional perspective due to their locality principle. For complex applications a powerful inscription language is necessary. Petri net formalisms often choose functional languages for this purpose to maintain the perspective. The Coloured Petri Nets formalism [8] for example uses ML as inscription language. To address the issue that simulators still have problems with side effects, the Curry-Coloured Petri Nets formalism uses a purely functional language to prevent side-effect related problems and logic program evaluation for the transition binding search [17]. In comparison to these formalisms, Reference nets allow the creation of dynamic hierarchies using the

---

**Figure 10: The Reference Net generated from the home automation DSML**

- **[cron.hourly]**
  - out0
  - [true]
  - [get a list of heating sensors]
  - [filter by office rooms]
  - [map temperature to heating level]
  - [configure heating]
  - [set temperature]

- **[new(roomfilter)]**
  - out0
  - [true]
  - [get a list of all rooms]
  - [configure heating]

- **[new(tempmap)]**
  - out0
  - [true]
  - [start heating]
  - [stop heating]

- **[new(setheating)]**
  - out0
  - [Set heating on/off]
Conclusion

Results

• Employ Reference Nets to model functional reactive systems and complex data structures
• RMT approach: development of DSML with functional component-based semantics
• Home automation DSML to control a heating system

Future work

• Formal analysis of DSML with functional semantics
• Petri net concepts for real higher-order functions
Bibliography

http://www.logos-verlag.de/cgi-bin/engbuchmid?isbn=2673&lng=eng&id=
http://www.sub.uni-hamburg.de/opus/volltexte/2010/4666/

https://doi.org/10.1109/ICSAW.2017.61

https://doi.org/10.1007/978-1-4842-4176-9_6


https://doi.org/10.1007/978-1-4615-1561-6
