Using Timed Input/Output Automata to Implement Distributed Systems

Chryssis Georgiou & Christos Ploutarchou
University of Cyprus

Peter M. Musial
EMC² and M.I.T.

Timed Input/Output Automata

Developing dependable distributed systems for modern computing platforms continues to be challenging. While the availability of distributed middleware makes feasible the construction of systems that run on distributed platforms, ensuring that the resulting systems satisfy specific safety, timing, and fault-tolerance requirements remains problematic. The middleware services used for constructing distributed systems are specified informally and without precise guarantees of efficiency, timing, scalability, composability, and fault tolerance.

Current software engineering practices limit the specification of such requirements to informal descriptions. When formal specifications are given, they are typically provided only for the system interfaces. Middleware interface syntax is usually strongly defined where computational semantics are often defined superficially. The specification of interfaces alone stops short of satisfying the needs of users of critical systems. Such systems need to be equipped with precise specifications of their semantics and guaranteed behavior. When a system is built of smaller components, it is important to specify the properties of the system in terms of the properties of its components.

We view formal specification and analysis as valuable tools that should be at the disposal of the developers of distributed systems. However, theoretically sound specifications have a limited impact; unless tools exist that automatically transform these specifications into executable code. Only if such tools are formally scrutinized, then the resulting executable code can be deemed as reliable and verifiably correct. This is the promise of our Tempo-to-Java translation module.

Motivation

ToDA provide natural mathematical notations for describing systems, have intended interfaces and the relationships between their descriptions at varying levels of abstraction.

A Timed IOA Automaton consists of:

- A set of states, including a nonempty subset of start states;
- A set of discrete labeled actions, classified as input, output, or internal;
- A transition relation, consisting of a set of (state; action; state) triples;
- A set of trajectories describing state evolution over time.

An action is enabled if its preconditions are satisfied; input actions are always enabled.

Tools features:

- System designer has the flexibility of using nondeterminism to allow multiple correct specifications of its system, hence relaxing assumptions on behaviors of the environment in which the system operates (at least at certain parts of the execution);
- Complex systems can be decomposed into sub-systems, where composition of these sub-systems yields the unified complex system. That enables one to view the specification at multiple levels of abstraction, and provides tractability between the top level system design, its intermediate evolution steps, and the final executable code.

TEMPO

TEMPO is a modeling framework for distributed systems with and without timing properties, where system models are described in the Tempo notation, derived from the TOA model, where systems are specified in terms of interacting timed automata.

The Tempo language closely matches semantics of the TOA framework and hence it inherits the rich set of capabilities for system modeling and analysis.

Using the Tempo Toolkit

1. Download the toolkit from [http://www.veromodo.com](http://www.veromodo.com)
2. Create a new project (Figure 1: File -> New -> Project)
3. Add a new model file to the project (Figure 2: Right-click on the project -> New -> File)
4. Pure and check for syntax errors (Figure 2)
5. Choose communication channel (Figure 3: Window -> Preferences -> Tempo Plugins -> Java generator)
6. Run translation process (Figure 6)

Translation Process

- Decide Communication Protocol (either MPI or TCP)
- Import and use auxiliary automata and vocabularies (based on selected protocol)
- Use the IDA compiler to compose automata to form a single automaton that describes all the computation, local to a single node in the system.
- Resolve nondeterminism by creating a Schedule
- Use TempoJava plug-in for translation

Translated Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI Mutual Exclusion Algorithm</td>
<td>All algorithms have been implemented in both MPI and TCP versions</td>
</tr>
<tr>
<td>Fully Leader Election Algorithm</td>
<td>Resulting translation tool has been successfully used and tested in the classroom setting at MIT and the University of Cyprus, at graduate level, and for senior year projects at undergraduate level</td>
</tr>
<tr>
<td>Distributed Clock Synchronization Algorithm</td>
<td></td>
</tr>
<tr>
<td>Phase Consensus Algorithm</td>
<td></td>
</tr>
<tr>
<td>Partial Reversal Algorithm</td>
<td></td>
</tr>
</tbody>
</table>

Resources