Elaborator and Runtime Library
- A Metropolis Backend Tool

Xi Chen, Guang Yang, Harry Hsieh, Felice Balarin and Yoshi Watanabe

Outline

• Elaborator and the elaboration process
  • Implementation
  • Network elaboration
  • Constraint elaboration
• Runtime library
  • What is runtime library
  • How to use runtime library
• Applications of elaborated constraints
  • Annotation trace generation
  • LOC checker generation
What is Elaborator?

- A backend tool that can be called by the user or other backend tools
- Input: MMM source code/abstract syntax trees
- Output: the structure of the network
  - Object nodes in the network (e.g. processes)
  - Connections between objects
  - Refinement hierarchy
  - Constraint instances (constraint elaboration)
  - Resolved runtime structural keywords, e.g. getconnectionnum, getconnections src, ..., etc

The Elaboration Process
Implementation of Elaborator

- Translate MMM objects (e.g. process, medium) to Java classes
- Top-level netlist is instantiated, and all the other objects are instantiated in turn
- Only constructors of the objects in the network are executed
- The network is built using the runtime library by executing the Java code
- Location: metropolis.metamodel.backends.elaborator

An Example of Network Elaboration

```java
public netlist IwIr {
    public IwIr(String name) {
        ...
        int numP = 2;
        for (int i = 0; i < numP; i++) {
            XX p = new XX("P" + i);
            addcomponent(p, this, "P" + i);
            connect(p, port1, m);
            connect(p, port0, r);
        }
        ...
    }
}
```

```
public class IwIr extends metamodel.lang.Netlist {
    public IwIr(String name) {
        super(name);
        ...
        int numP = 2;
        for (int i = 0; i < numP; i++) {
            XX p = new XX("P" + i);
            Network.net.addComponent(p, this, "P" + i);
            Network.net.connect(p, "port1", m);
            Network.net.connect(p, "port0", r);
        }
        ...
    }
}
```

MMM Source Code | Java Code
An Example of Network Elaboration (cont’d)

Network structure generated by Java execution and represented by runtime library classes

An Example of Constraint Elaboration

• If m = 2, there are actually 2 different constraint instances

MMM Source Code

Java Code
An Example of Constraint Elaboration (cont’d)

netlist test.sumnet {
  o Instance name: top_level_netlist
  o Component name: null
  o Components:
    ...
    ...
  o Not refined by a netlist
  o Does not refine any node
  o Constraints:
    - LOC Constraint (# 0)
      o Container: top_level_netlist
      o Event references:
        - beg(datagen1, y2bf1.tokenLabel)
        - beg(sum1, bf2y1.tokenLabel)
    - LOC Constraint (# 1)
      o Container: top_level_netlist
      o Event references:
        - beg(datagen1, y2bf1.tokenLabel)
        - beg(sum1, bf2y1.tokenLabel)
  }

*** List of annotations ***
  o beg(sum1, bf2y1.tokenLabel) k[0]
  o beg(sum1, bf2y1.tokenLabel) k[1]

The print-out of the elaborated constraints

Advantages of Elaboration

- Get the network structure before doing anything else
- Resolve runtime keywords or variables
- Useful to many other backend tools
  - Simulation – SystemC
  - Verification – Promela
  - Constraint monitoring or checking
    ... etc
**How to Use Elaborator**

- Elaborated network is normally utilized by other backend tools
- Call elaborator and get the elaborated network
- Use runtime library API to access and manipulate the elaborated network
- Example: SystemCBackend class is defined as a subclass of ElaboratorBackend class

**Runtime Library**

- Represent and manipulate the elaborated network structure
- A set of Java classes located in metropolis.metamodel.runtime
- Java classes in runtime library:
  - Network – describe the whole elaborated network
  - MMType – specify a particular node type, e.g. a process type or a netlist type
  - INode – represent an object node, e.g. a medium instance
  - INetlist – represent an object of netlist, e.g. a netlist instance
Runtime Library (cont’d)

- More Java classes in runtime library:
  - MMPort – specify a port type
  - IPort – represent a port instance
  - Connection – specify a connection between 2 nodes through ports
  - Event – represent an event reference, e.g. beg(process, medium.label)
  - Constraint – represent a constraint instance

Runtime Library (cont’d)

- The network structure can be accessed by calling runtime library APIs, for example:
  - Network.getNodes() – get a list of nodes in the network
  - Network.getNetlist() – get a particular netlist by name
  - Network.show() – return a string that describes the network

- The network structure can also be modified by calling runtime library APIs, for example:
  - Network.flatten() – flatten the elaborated network into a network where refined nodes and connections are replaced by their refinements
LOC Constraints in MMM

- LOC is a transaction-level quantitative constraint language
- Directly supported by MMM syntax
- Using MMM keywords constraint and loc
- For example (a latency constraint):

```c
constraint {
    event P0_start = beg(p0, p0.start);
    event P0_finish = beg(p0, p0.finish);
    loc(forall (int i) t@(P0_finish,i) - t@(P0_start, i) <= 20 );
}
```

Annotation Trace Generation

- An application of elaborated constraints
- Utilize elaborated constraints and annotations
- Trace generation – insert “print” statements into SystemC code
An Example of Annotation Trace Generation

```
constraint {
  event P1_start = beg(p1, p1.start);
  event C_start = beg(c, c.start);
  loc(forall (int i) w@(P1_start,i) == w@(C_start, i) );
}
```

A Constraint in MMM

- `beg(p1, p1.start)`
- `beg(c, c.start)`

List of annotations

- `beg(p1, p1.start)` w
- `beg(c, c.start)` w

Elaborated annotations

SystemC Backend

SystemC Simulation w/ trace generation

Trace from SystemC Simulation

LOC Checker Generation

- Another example of elaborated constraints
- Utilize elaborated constraints and annotations
- We are still working on it
A Complete Example of LOC Checking

```java
public class IwIr extends metamodel.lang.Netlist {
    public IwIr(String name) {
        /*constraint block*/
        Constraint __tmpConstraint;
        Event P1_start = new Event(Event.BEG, p1, p1, "start");
        Event C_start = new Event(Event.BEG, c, c, "start");
        // loc( forall(int i) w@(P1_start,i) == r@(C_start, i+1) );
        tmpConstraint = new Constraint(Constraint.LOC);
        Network.net.getNode(this).addConstraint(__tmpConstraint);
        tmpConstraint.addEvent(P1_start);
        Network.net.addAnnotation(P1_start, "w");
        tmpConstraint.addEvent(C_start);
        Network.net.addAnnotation(C_start, " w");
    }
    ...
    }
}
```
To Be Done

• Integrate LOC monitors into SystemC simulation
• Resolve runtime structural keywords, e.g. getconnectionnum, getconnections, ...
• Check or monitor LTL constraints