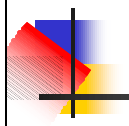
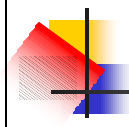


Metropolis Verification Backends

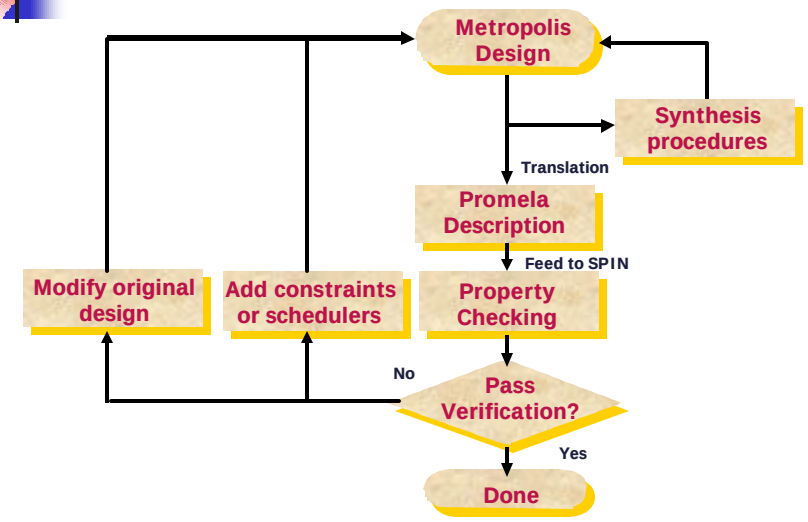
Xi Chen, Guang Yang, Harry Hsieh,
Felice Balarin and Yoshi Watanabe



Outline

- Promela Backend in Metropolis
  - Formal Verification Methodology
  - Model Checker SPIN
  - Implementation
  - Case Study
- Constraint Checking for Simulation in Metropolis
  - Constraint elaboration
  - Annotation trace generation
  - LOC trace checker generation

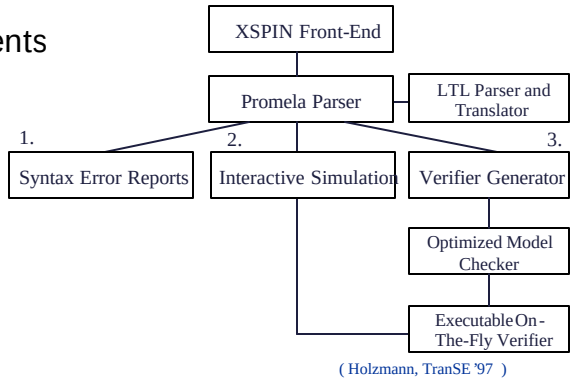
Formal Verification Methodology



3

Model Checker SPIN and Promela

- Promela language elements
  - Concurrent process
  - Communication channels
  - Atomic statement
  - Flow control statements
  - Case selection statements



(Holzmann, TranSE '97)

- **Promela is a C-style procedural format with simple constructs suitable for protocol verification**

4

Translation from MMM to Promela

- Processes/media

  →Proctypes with functional inlining

- Dynamic objects, e.g. dynamic arrays

  →Restricted and static objects, e.g. static arrays

- Function-architecture mapping

  →Using rendezvous channels to synchronize functional processes and mapping processes

5

Translation of "await"

await {

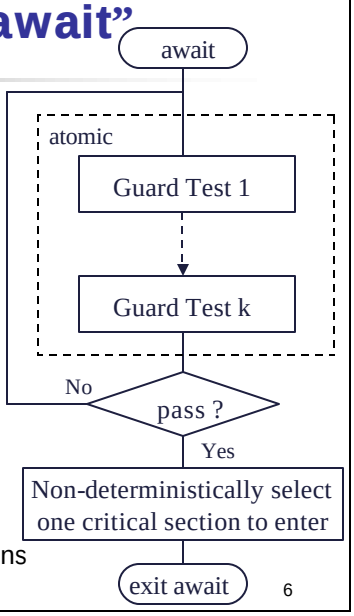
($guard_1$; $testlist_1$; $setlists_1$) {$stmts_1$}

($guard_k$; $testlist_k$; $setlists_k$) {$stmts_k$}

}

"await" statement

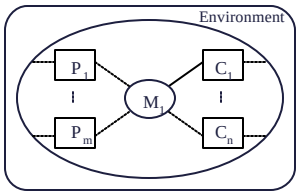
- Synchronize concurrent processes
- Multiple critical sections
- Guard and semaphores
- Non-deterministic selection of critical sections



6

Case Study – Producer/Consumer

Property: *"Whenever the producer starts to write an item into the medium, there must be some space in the medium"*



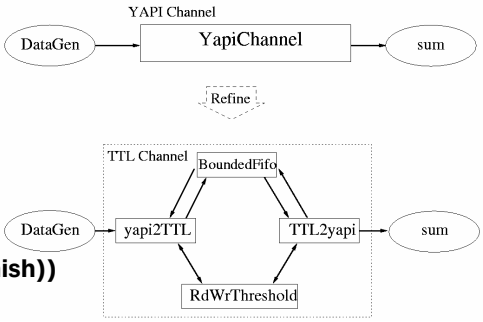
A typical producer- consumer network

LTL: $\bullet((P_1\_write \vee \ldots \vee P_m\_write) \rightarrow M_1\_not\_full)$

- When m = 2, n = 1 and M_1 has single space, source code: 120 lines, Promela code: 650 lines. Verification is passed within 1s CPU time and 13MB memory.

7

Case Study – YAPI and TTL



To check if there is a deadlock situation within the YAPI and TTL channel, use:

"once the writer starts writing data into the channel, it will finish it eventually".

- **(datagen_start → (<> datagen_finish))**

- This property is verified on the YAPI level with exhaustive verification within 9 hours.
- A deadlock situation is found in the TTL channel.
- After the bug in TTL channel fixed, the property is passed within 4 hours.

8

Future Work

- Safe abstraction

- Automatic Abstraction propagation in Metropolis

Automatically apply the abstractions throughout the entire design according to initial abstraction specifications

9

Outline

- Promela Backend in Metropolis
  - Formal Verification Methodology
  - Model Checker SPIN
  - Implementation
  - Case Study
- Constraint Checking for Simulation in Metropolis
  - Constraint elaboration
  - Annotation trace generation
  - LOC trace checker generation

10

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):

```
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 );
}
```

- Constraints can be extracted and instantiated during network elaboration

Constraint Elaboration – An Example

```
public netlist sumnet {
  …
  constraint{
    event Wevent = beg (…);
    event Revent = end (…);
    for(j = 0; j < m; j ++)
      loc(forall (int i) k[j]@(Wevent,i) == k[j]@(Revent,i));
  }
  …
}
```

MMM Source Code

```
public class sumnet extends metamodel.lang.Netlist {
  public IwIr(String name) {
  …
  /*constraint block*/ {
    Constraint __tmpConstraint;
    Event Wevent = new Event(…);
    Event Revent = new Event(…));
    for(j = 0; j < m; j ++) {
        // loc(forall (int i) k[j]@(Wevent,i) == k[j]@(Revent,i));
        tmpConstraint = new Constraint(Constraint.LOC);
        Network.net.getNode(this).addConstraint(__tmpConstraint);
        tmpConstraint.addEvent(Wevent );
        Network.net.addAnnotation(Wevent ,"k[" + i + "]");
        tmpConstraint.addEvent(C_start);
        Network.net.addAnnotation(Revent ,"k[" + i + "]");
  }
  …
}}
```

Java Code

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

Constraint Elaboration – An Example (cont'd)

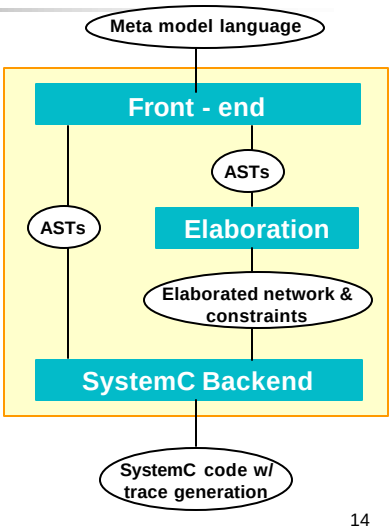
```
netlist test.sumnet {
  o Instance name: top_level_netlist
  o component name: null
  o Components:
       …
       …
  o Not refined by a  netlist
  o Does not refine any node
  o Constraints:
       - LOC Constraint (# 0)
         o Container: top_level_netlist
         o Event references:
   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

The print-out of the elaborated constraints

- Constraints are indexed in an node
- Event references are saved
- A list of annotations are saved in the network

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

```
*** List of annotations ***
  o beg(p1, p1.start)  w
  o beg(c, c.start)  w
```

Elaborated annotations

SystemC Backend

SystemC Simulation w/ trace generation

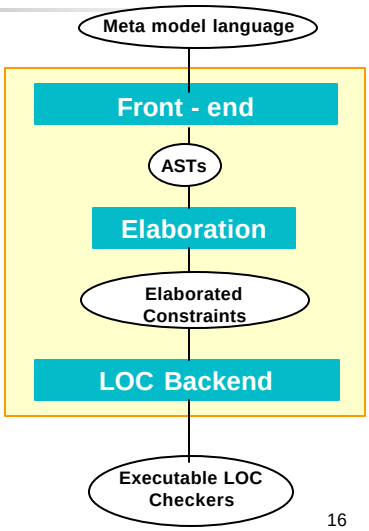
```
BEG_Consumer_Consumer_start 0
BEG_Producer1_Producer1_start 0
BEG_Producer1_Producer1_start 1
BEG_Producer1_Producer1_start 2
BEG_Consumer_Consumer_start 1
BEG_Producer1_Producer1_start 3
BEG_Consumer_Consumer_start 2
BEG_Producer1_Producer1_start 4
BEG_Consumer_Consumer_start 3
⋮
```

Trace from SystemC Simulation

15

LOC Checker Generation

- Another utilization of elaborated constraints
- Generate checkers from elaborated constraints and annotations



16

A Complete Example of LOC Checking

```
public netlist IwIr {
  public IwIr(String name) {
    …
   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) );
   }
   …
  }
}
```

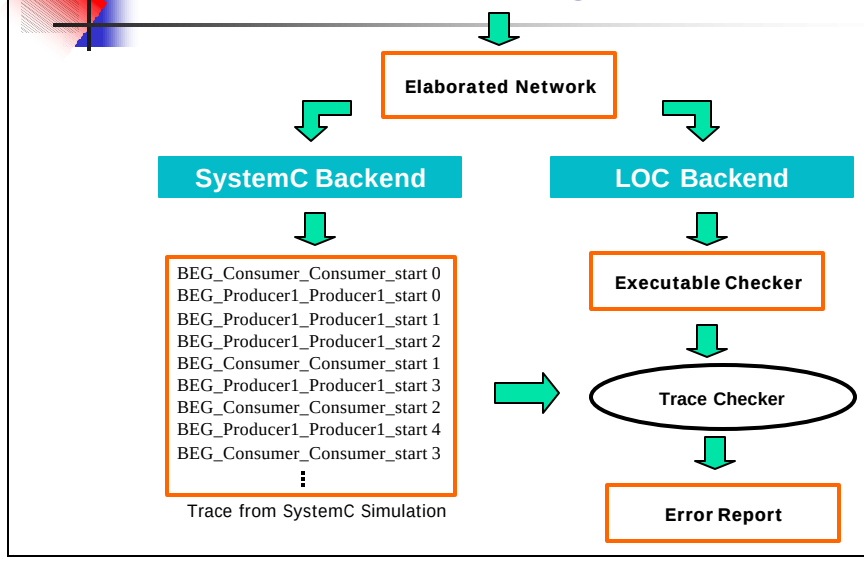
MMM Source Code

```
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");
  }
  …
}}
```

Java Code

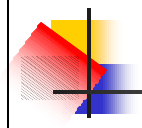
17

A Complete Example of LOC Checking



Trace from SystemC Simulation

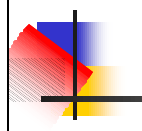
18



Future Work

- Integrate LOC monitors into SystemC simulation
- Check or monitor LTL constraints

19



Thank you!

20