A step towards reconciling GALS industrial design with formal verification

Fatma Jebali
Join work with Frédéric Lang & Radu Mateescu
Inria Grenoble – France

LASER Summer School
September 13th, 2014
GALS: Globally Asynchronous, Locally Synchronous

Synchrony
- Several components governed by a single clock
- Instantaneous computations and communications
- Deterministic behaviour

Asynchrony
- Several subsystems executing at different speeds and interacting asynchronously
- Communications take arbitrary delays
- Nondeterministic behaviour
Em4: smart generation of PLCs (Programmable Logic Controllers)

- Software-based and multi-platform systems
- Easy-to-use programming tool (block diagram model)
- Application-oriented programming

Vehicle control access, lighting control, security units, ...

Mr. Schneider
Em4: smart generation of PLCs (Programmable Logic Controllers)

Want my em4 be smarter:
- let distributed programs communicate with each other
- program and validate applications in the cloud from different mobile platforms

Seems to be innovative. **But**, how can I validate em4 networks?

Mr. Schneider
Verification of Em4 applications

Several platforms?!

Powerful automatic tools
Correct systems
But, expertise required:
Concurrency, formal methods

How to bridge the gap between industrial design software and formal verification tools?

Several platforms?!

Mr. Synch

Mr. Schneider

Mr. Asynch

Synchronous features?!
**CADP** _Construction and Analysis of Distributed Processes_

- Modular toolbox based on formal methods (nearly 50 tools up to now)

- Explicit-state verification
  - Model checking (μ-calculus, MCL)
  - Equivalence checking (bisimulations)
  - Visual checking

- Different techniques
  - Exhaustive
  - Partial
  - On the fly
  - Compositional
  - Distributed

[http://cadp.inria.fr](http://cadp.inria.fr)
The challenge

- Scalability of Model-checking (state space explosion)
- Expertise required to ensure correct and efficient specifications, write properties
- Automatic modeling and verification is recommended
  - Ability to handle industrial-size applications
  - Designer-friendly interfaces
Towards industrial-friendly formal verification

- CADP Toolbox
- Evaluator
- Interactive interface
- Schneider designer
- MCL generation
- LNT generation
- ...2mcl
- grl2Int
- Properties
- LNT
- Caesar
- Verification results
- Hand-written
- Automatically generated
Towards industrial-friendly formal verification
Blocks: synchronous programs

```
block Controller1(out Light : bool) {
    {receive Temperature : nat} is

    allocate Block_Comparator [Superior] as B02_Comparator,
    Block_Numeric_Constant [Threshold2] as B03_Numeric_Constant,
    Block_Xw_In as B07_Xw_In

    perm c0 : nat
    temp c1 : nat, c2 : nat
    B07_Xw_In (Temperature, ?c1);
    B03_Numeric_Constant (?c2);
    B02_Comparator (_; c1, c2; ?Light)

end block
```
Blocks as reactive systems

- Permanently interaction with the environment
- Constraints on one block
- Constraints on several blocks
Blocks as reactive systems

```plaintext
environment Env_Sensor (out Sensor : nat) is

  on ?Sensor -> Sensor := any nat
  where ((Sensor <= 3)
    and (Sensor >= 1))

end environment

environment Env_Sensor (out Sensor : nat) is

  on ?Sensor -> Sensor := select
    Sensor = 1
    []
    Sensor = 2
    []
    Sensor = 3
    end select

end environment
```
Blocks as asynchronously communicating systems

Accurate design of

- complex network topologies (bus, ring, star, etc.)
- connection modes (point-to-point, multi-point, etc.)
- communication protocol

Existing languages

- rigid topologies
- point-to-point communications between separate synchronous systems
Blocks as asynchronously communicating systems

```
medium Med {receive Input : nat | send Output : nat} is
    perm Buffer : nat := 0
    select
        on Input -> if (Buffer == 0) then
            Buffer := Input
        else
            null
        end if
    end select
[]
on ?Output -> if (Buffer == 0) then
    Output := 0
else
    Output := Buffer;
    Buffer := 0
end if
end select
end medium
```
Asynchronous composition and communication

- Define the system actors: blocks, environments, mediums
- Specify interactions between actors (message-passing synchronizations)
- Blocks execute arbitrarily and cyclically (active behaviour), triggering the activation of connected environments and mediums (passive behaviour)
Asynchronous composition and communication

```
system Main (in Switch : bool; in Sensor : nat;
               out Is_On : bool; out Light : bool) is

  allocate Diagram1 as Controller1, Diagram2 as Controller2,
                  Med as Medium1,
                  Env_Sensor as Env1

  temp Send_Temperature : nat, Receive_Temperature : nat

network
  Controller1 (Switch; Sensor; ?Is_On1) {?Send_Temperature},
  Controller2 (?Light) {Receive_Temperature}
constrainedby
  Env1 (?Sensor)
connectedby
  Medium1 {Send_Temperature | ?Receive_Temperature}
end system
```
This is just the tip of the iceberg...
First results


Automatic translator grl2lnt already designed and implemented, paper to be submitted (ETAPS 2015)
Ongoing work

- Automatic generation of GRL code from em4 software applications (large-scale industrial case studies)
- Industrial-friendly property language

Future work

- Automatic C code generation from GRL
- Connect GRL to other industrial frameworks to handle more instances of GALS systems
- Connect GRL to other verification tools
Thank you