# Foundations Software Technology ($F_aST$)

Farhad Arbab

**LIACS**

*Leiden Institute of Advanced Computer Science*

Bachelorklas December 3, 2013

Universiteit Leiden
The Netherlands

# FaST Members

Joost Kok (also Algorithms)

Farhad Arbab (CWI)
Frank de Boer (CWI)

Grzegorz Rozenberg (also Alg)

Marcello Bonsangue (also CWI)
Michel Chaudron
Luuk Groenewegen
Jetty Kleijn

PostDoc
Natallia Kokash

PhD Students
Stijn de Gouw
Michiel Helvenstein
Sung-Shik Jongmans
Jurriaan Rot
Pieter Kwantes (ext)
Narges Khakpour (ext)
Behnaz Changizi (ext)
B. Pourvatan (ext)
J. Winter (ext)
…

# FaST

- Formal semantic foundations of software composition and coordination

- Software Engineering

- Theoretical Computer Science

# FaST

- Focus on the formal semantic foundations of software composition and coordination:

- Large *software systems* are difficult to construct and maintain due to their inherent complexity.
- *Compositional* techniques hold the key to breaking this complexity down to manageable levels.
- Composing systems out of *independent components* or services requires *coordination* of their interactions.
- Considerations for *concurrency, distribution, mobility, and dynamic reconfiguration* of systems, e.g., to upgrade or adapt to their changing environment, add to the complexity of a system and its interaction protocols.
- Coordination in Software Systems studies how complex systems can be constructed from independent components or services using a clear distinction between individual components or services, and the protocols for their coordinated interaction.

# Mission

- Development of formalisms, methods, techniques, and tools to design, analyze, and construct software systems out of components and services.

- Ingredients
  - Classes/Objects
  - Components
  - Services
- Construction
  - Composition
  - Correctness
- Issues
  - Concurrency
  - Coordination
  - Model
- Approach
  - Formal methods
  - Experimental systems
  - Empirical studies

# Areas

- Formal models of concurrent, distributed, object oriented, and component-based systems
- Formal semantics, process algebras and logics for reasoning about such systems
- Dynamically reconfigurable adaptable systems
- Concurrency on multi- and many-core platforms
- Testing, deductive verification, and model checking
- Software services and cloud computing
- Quality of service
- Empirical studies of the effects of methods and techniques on productivity and quality of industrial software development projects.

# Major challenge

Development of techniques for effectively establishing behavioral properties of dynamical systems

# Activities – F. Arbab

- Coordination models and languages
  - Coordinated composition of software intensive systems
  - Concurrency and interaction
  - Coordination language Reo
  - Constraint automata

- Use of coordination
  - Compositional QoS
  - Code generation for multi-core systems
  - Service oriented computing
  - Testing

# Activities – F.S. de Boer

- Software correctness

Programming logics
- Deductive proof methods for the verification of programs

Object Orientation
- Verification and Testing

Concurrency
- Semantics

Integrated Formal Methods
- Testing
- Model Checking
- Deductive Verification
- Abstraction

# Activities – M. Bonsangue

- Formal Methods

  Mathematically-based techniques for the specification, development and verification of software and hardware systems

  - Testing object-oriented languages
  - Semantics and model checking of software connectors
  - Semantics and verification of dynamical evolving systems

- Algebra, Coalgebra and Logic

  Mathematical frameworks for the specification of the reactive behaviour of systems

  - Process algebra, regular expressions
  - (Probabilistic, non-deterministic, …) automata
  - Modal logics

# Activities – J. Kleijn

- Formal Methods

Mathematical specification (of distributed behaviour)

- Automata and languages
- Concurrency Monoids (generalised traces)
- Extended partial orders

- Modeling Concurrency

Petri Nets

- Theory
- Biologically motivated models
  - membrane systems, reaction systems: analysis and synthesis
- Biomodeling

Team Automata

- cooperating components

Application:

- Financial/business processes

# Current projects:

- 25 different projects

# Parallelism toolkit on Kalray processor

- Problem: Modify the proto-runtime toolkit so that it operates efficiently on the Kalray ultra low power, high performance embedded processor.
- Required: Programming experience in C and assembly.
- Supervisors: Sean Halle and F. Arbab

# Proto-runtime + V8 Javascript interpreter

- Problem: Connect the proto-runtime toolkit to the high performance V8 javascript interpreter.
- Required: Programming experience in C and Javascript.
- Supervisors: Sean Halle and F. Arbab

# Proto-runtime on Dutch National Supercomputer

- Problem: Measure the performance of proto-runtime on the Netherland's national supercomputer, and experiment with ways to improve its performance.
- Required: Programming experience in C and assembly.
- Supervisors: Sean Halle and F. Arbab

# Implement your own parallel language

- Problem: Use the Rascal meta environment to write a simple source-to-source translator that translates your custom parallel syntax into calls to a parallel runtime system.
- Required: C programming on Linux
- Supervisors: Sean Halle and F. Arbab

# Eclipse GUI to run Reo applications

- Problem: Develop an Eclipse GUI for the ECT to produce the executable files for a concurrent application. This tool links various components and/or threads with the Reo coordinator code, produces a main program, and supplies it with its command-line parameters.
- Required: Java programming and Eclipse
- Supervisors: S-S Jongmans and F. Arbab

# An editor for data constraints

- Problem: Develop a basic editor (syntax highlighting, useful error messages, basic refactoring, etc.) for a simple constraint language in Eclipse, by using---and if necessary extending---an existing parser.
- Required: Java programming; (willingness to learn) Eclipse and Antlr

Supervisors: S-S Jongmans and F. Arbab

# Web service deployment in the cloud

- Problem: Develop a Java library for deploying web services, as jar files, on Amazon EC2 virtual machines in the cloud, by using functionality from Amazon's AWS Toolkit for Eclipse.
- Required: Java programming; (willingness to learn) Eclipse, JAX-WS, AWS, AWS Toolkit for Eclipse
- Supervisors: S-S Jongmans and F. Arbab

# Test cases generation for Java classes

- Problem: Given a piece of Java code generate inputs so to test it “enough”.
- Required: Knowledge of Java + P&C.
- Supervisors: F. de Boer and M. Bonsangue.

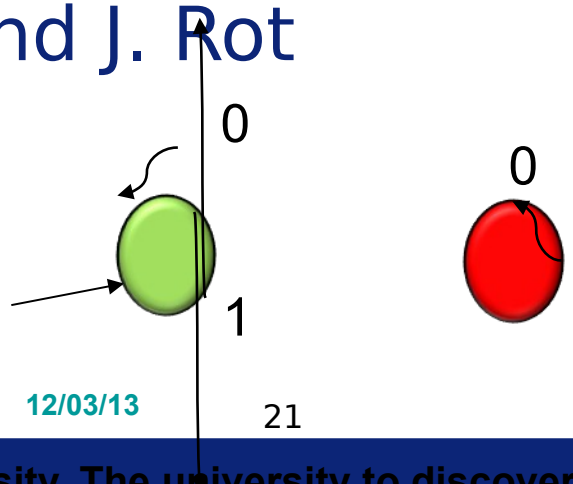
```
while ( low <= high ) {
    int mid = (low + high ) / 2
}
```

# Automata toolkit

- Problem: Implement novel techniques to
  Minimize automata
  Solve decision problems
  (language equivalence, minimality, inclusion)
  Go from regular expression to automaton and back

- Required: knowledge of automata theory (FI2),

- Supervisors: M. Bonsangue and J. Rot



# Model checking recursive programs

- Problem: Recursive programs over finite data structures may have infinite state. Implement a way to check all of them against a property.

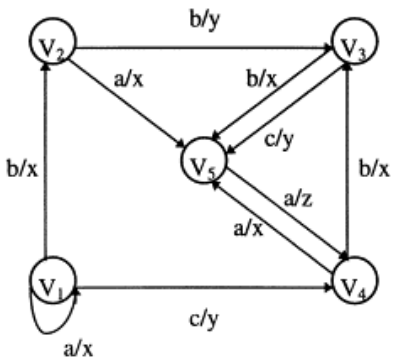
- Required: knowledge of automata theory (FI2), logic and P&C

- Supervisors: F. de Boer, M. Bonsangue and J. Rot

```
Proc P ==  .... call Q ...
Proc Q == ... call P ....
```

# Process equivalence checking

- Problem: Processes are abstract description of the behaviours of systems. They are subject to several equivalences: bisimulation, failure, ready, ... Implement recent techniques to check equivalent processes.
- Required: knowledge of automata theory (FI2).
- Supervisors: M. Bonsangue



# KAT partial derivatives

- Problem: Develop the theory to associate a non-dtereministic finite automaton to a KAT expressions using partial derivatives.
- Required: knowledge of formal languages and automata theory (FI2).
- Supervisors: M. Bonsangue and Jurriaan Rot

# Parsing Boolean grammars

- Problem: Develop parsing technique for the Boolean grammar, and syntactic formats so to characterize regular languages.
- Required: knowledge of formal languages and automata theory (FI2).
- Supervisors: M. Bonsangue and Jurriaan Rot

# Music from streams

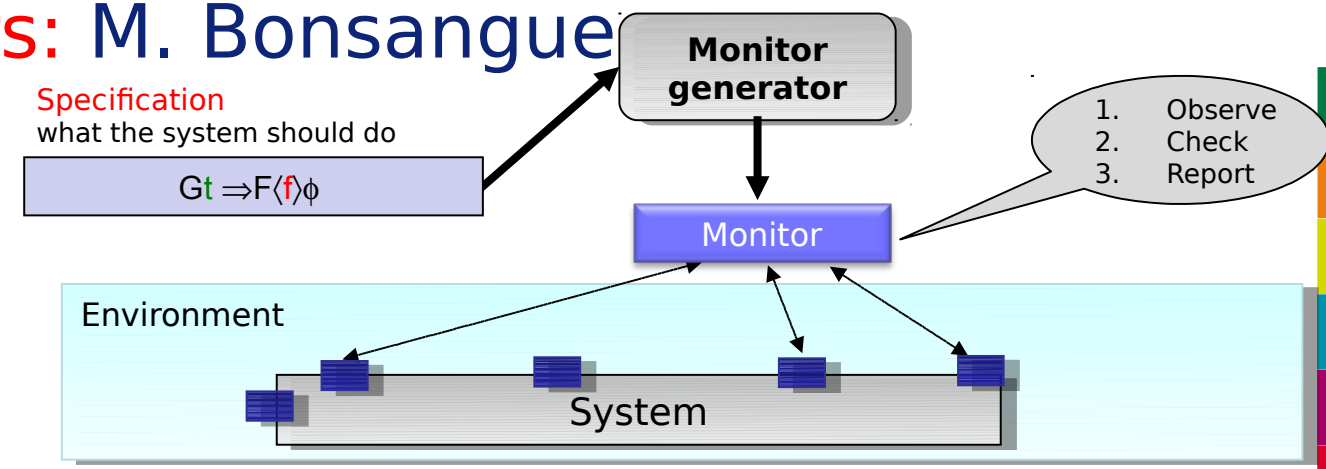
- Problem: Build a system to transform stream definitions into musical note. The idea is to generate interactively a midi file from the number of the stream defined by an user.
- Required: Programming in other languages than C++
- Supervisors: M. Bonsangue and Jurriaan Rot

# Bisimulation-up-to for formal languages

- Problem: Implement several bisimualtion-up-to techniques for language inclusion and equivalence.
- Required: Programming in other languages than C++; Running experiments
- Supervisors: M. Bonsangue and Jurriaan Rot

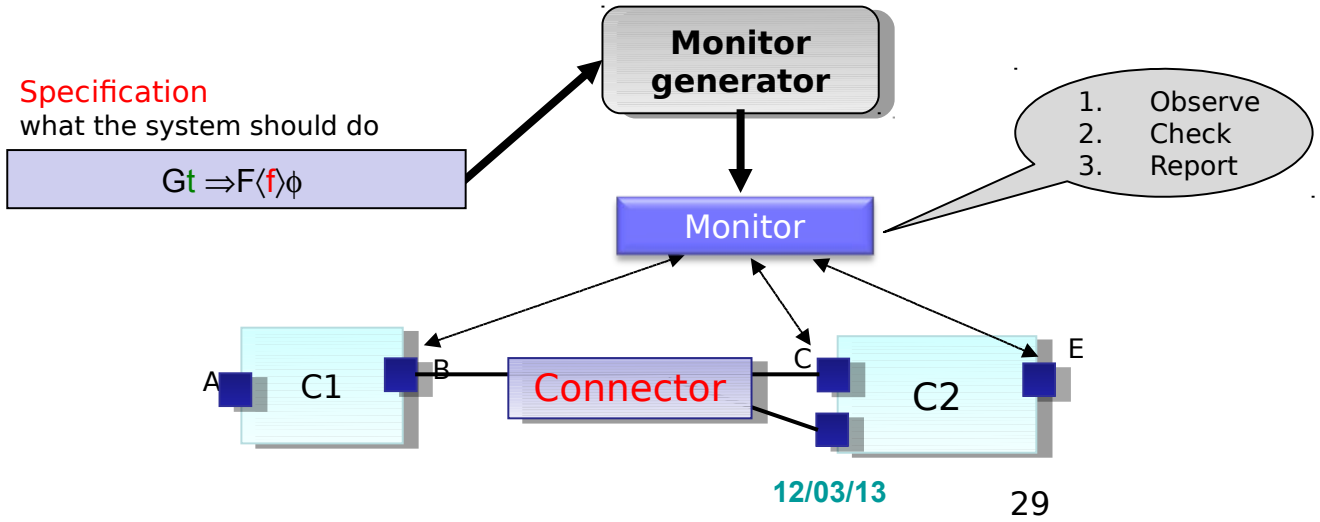
# Monitoring circuits

- Problem: Automatic generation of monitors for simple hardware circuits to check their behaviors at run time.
- Required: Knowledge of Logic and automata theory
- Supervisors: M. Bonsangue



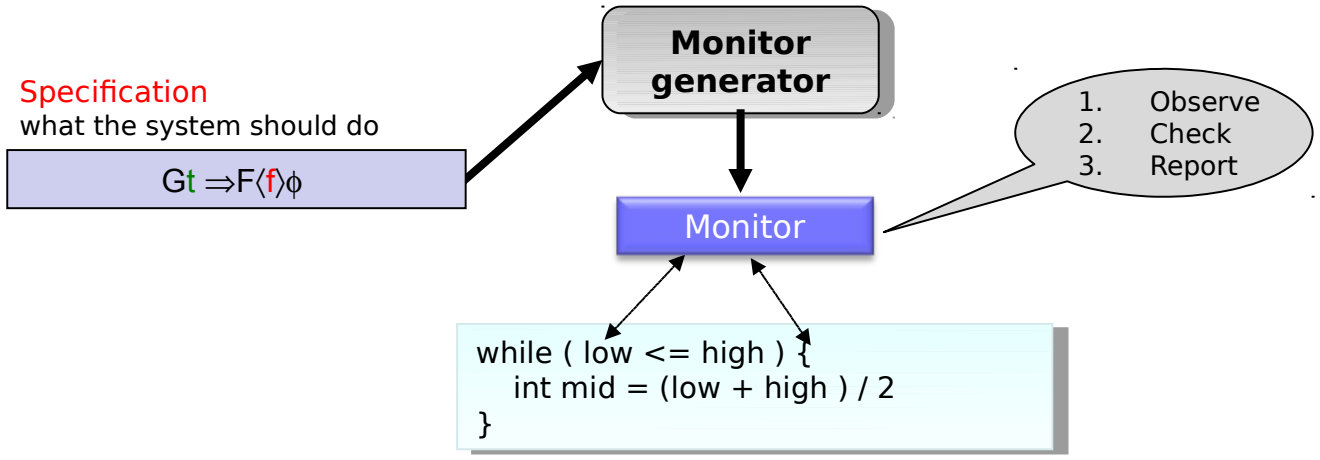
# Monitoring sofware circuits

- Problem: Same as previous one, but now with coordination software instead of circuits.
- Required: Knowledge of Java
- Supervisors: F. Arbab and M. Bonsangue



# Monitoring Java

- Problem: Same as previous one, but now with Java.
- Required: Knowledge of Java
- Supervisors: F. de Boer and M. Bonsangue



# Coverability

- Problem: Overview of algorithms and constructions for boundedness, finiteness and coverability properties for different classes of Petri Nets.

- Required: Theory of concurrency, algorithmic interest.

- Supervisors: J. Kleijn

# Compatibility of Teams

- Problem: Implement compatibility checks for team automata: can components collaborate successfully?

- Required: theory of concurrency, automata

- Supervisors: J. Kleijn

# Bio inspired Modeling

- Problem: Set Nets, a Petri net model for reaction systems. Relation to classical net models.
- Requirements: Theory of Concurrency.
- Supervisors: J. Kleijn

# Bio Modeling

- various issues:

- Petri nets as operational models for real-life phenomena
- Feature modeling

- Tools:

- -Requirements: Theory of Concurrency.

- Supervisors: J. Kleijn and F.Verbeek

# Financial product Markup Language

- Problem: The development of the domain specific language FpML: design and challenges
- Required: Informatica en Economie
- Supervisors: P. Kwantes and J. Kleijn

# Business Process Modeling Notation

- Problem: The ontological adequacy of BMPN
- Required: Informatica en Economie
- Supervisors: P.Kwantes, J.Kleijn/F.Verbeek

# Treemaps

- Problem: Multi-scale visualisation of maps
- Required: Datastructures, SE, Imaging (?)
- Supervisors: N. Kokash and J.Kok