Collaboration Notes: Prof. Marcello Bonsangue

Scientific Director - Systems Modelling & Analysis

Background Research

Key Publications & Recent Work

1. **Simulating Quantum Circuits by Model Counting** (CAV 2024)
   - With Jingyi Mei and Alfons Laarman
   - Novel approach using model counting for quantum circuit simulation
   - Bridges classical verification techniques with quantum computing
2. **From Symbolic Constraint Automata to Promela** (2022)
   - Published in Journal of Logical and Algebraic Methods in Programming
   - Translates constraint automata to executable Promela code for SPIN model checker
   - Demonstrates practical application of theoretical automata work
3. **Coalgebraic Methods** (Long-term research theme)
   - Pioneer in applying coalgebra to computer science
   - Focus on automata, bisimulation, and formal semantics
   - Over 6,600 citations, indicating significant impact

Core Expertise Areas

- **Coalgebra**: Mathematical framework dual to algebra
- **Formal Methods**: Verification, model checking, automata theory
- **Quantum Computing**: Recent focus on quantum circuit verification
- **Symbolic Constraint Automata**: Extension of traditional automata with data constraints

Understanding Coalgebra (For Your Preparation)

What is Coalgebra?

Coalgebra is the mathematical dual of algebra. While algebra focuses on constructing complex objects from simpler ones, coalgebra focuses on observing and decomposing complex objects.

Key Concepts:

1. **States and Observations**: In coalgebra, we have a state space X and a function f: X → F(X) that describes observations and transitions
2. **Functor F**: Determines the type of system (deterministic, probabilistic, etc.)

3. **Bisimulation**: The fundamental equivalence notion - two states are bisimilar if they exhibit the same observable behavior

Why It Matters for APIs:

- APIs are essentially state machines with observable behaviors
- Coalgebraic methods provide a mathematical foundation for reasoning about API behavior
- Bisimulation can formalize when two API implementations are equivalent

Symbolic Constraint Automata

These extend regular automata by:

- Adding data constraints on transitions
- Incorporating local variables
- Enabling reasoning about both control flow AND data flow

Connection to jGuard:

- jGuard guards are essentially state variables
- Requirements/consequences form transition constraints
- Your work could be formalized coalgebraically!

Concrete Collaboration Ideas

Project 1: Coalgebraic Formalization of jGuard

Goal: Provide mathematical foundations for jGuard using coalgebra

Technical Approach:

1. Model jGuard-annotated APIs as coalgebras where:
   - State space = guard valuations
   - Transitions = method calls with requirements/consequences
   - Observations = API behavior
2. Define bisimulation for jGuard APIs:
   - Two implementations are equivalent if they enforce the same usage patterns
   - This could lead to API refactoring tools
3. Use his symbolic constraint automata work:
   - Guards become constraint variables
   - Requirements become transition constraints
   - Meta-variables fit naturally as parameters

Benefits:

- Formal correctness proofs for jGuard
- Connection to model checking tools (via his Promela work)
- Publication in top venues (CAV, LICS, etc.)

Project 2: Quantum-Enhanced API Analysis

Goal: Apply quantum algorithms to large-scale API misuse detection

Technical Approach:

1. Leverage his CAV 2024 quantum circuit work
2. Formulate API misuse detection as a model counting problem
3. Use quantum algorithms for exponential speedup on certain patterns

Specific Ideas:

- Quantum algorithms for finding API usage patterns in massive codebases
- Grover's algorithm for searching misuse patterns
- Quantum annealing for optimal API design

Project 3: Verified Synthesis of API Contracts

Goal: Automatically generate jGuard specifications from examples

Technical Approach:

1. Use coalgebraic learning techniques
2. Given positive/negative API usage examples
3. Synthesize minimal jGuard contract that accepts good uses, rejects bad ones

Connection to His Work:

- Builds on his automata learning research
- Uses symbolic constraint framework
- Could integrate with SPIN for verification

How to Present Your Ideas

Opening:

"I've been studying your work on coalgebraic methods and symbolic constraint automata. I see deep connections with my jGuard framework..."

Key Points to Emphasize:

1. **Mathematical Rigor**: Show you understand the theoretical foundations
2. **Practical Impact**: jGuard provides real-world application of his theories
3. **Novel Contributions**: Your work extends his ideas in new directions

Technical Terms to Use Correctly:

- **Coalgebra**: "jGuard specifications form a coalgebra where guards represent the state space"
- **Bisimulation**: "We could define when two API implementations are behaviorally equivalent"
- **Symbolic Constraints**: "Requirements in jGuard are essentially symbolic constraints on transitions"
- **Model Counting**: "API misuse detection reduces to counting satisfying assignments"

Questions to Ask Him

1. "How would you formalize the notion of API behavioral equivalence coalgebraically?"
2. "Could your quantum circuit verification techniques apply to API verification?"
3. "What's the relationship between symbolic constraint automata and runtime monitors?"

Potential Joint Papers

1. **"A Coalgebraic Semantics for Misuse-Resilient APIs"**
   - Venue: LICS or FoSSaCS
   - Content: Formal foundations of jGuard
2. **"Quantum Algorithms for API Misuse Detection"**
   - Venue: CAV or TACAS
   - Content: Scaling verification using quantum computing
3. **"From jGuard to SPIN: Verified API Contracts"**
   - Venue: TACAS or FM
   - Content: Tool paper connecting jGuard to model checking

Key Advantages of Collaboration

1. **Complementary Expertise**: You bring practical API safety, he brings theoretical foundations
2. **High-Impact Publications**: His track record ensures top venues
3. **Access to Resources**: Systems Modelling Lab, quantum computing connections
4. **PhD Students**: Could co-supervise students working on formal API verification

References for Deep Dive

1. Rutten, J.J.M.M. (2000). "Universal coalgebra: a theory of systems"
2. Jacobs, B. (2016). "Introduction to Coalgebra: Towards Mathematics of States and Observation"

3. Bonsangue's CMCS workshop proceedings (editor) - shows breadth of coalgebra applications

Final Tips

- Be prepared to write on the whiteboard - coalgebraists love diagrams
- Show enthusiasm for the mathematical beauty, not just practical applications
- Mention specific papers of his you've read
- Be ready to explain jGuard formally, possibly using category theory notation