Distributed Control of Mobile Agents with Unreliable Communication: Theory & Experience with S5 Tools

Concetta Pilotto

K. Mani Chandy

Funded by AFOSR-MURI

Computer Science Department

California Institute of Technology

Outline

- Reliable Multi-Agent Systems (MAS) with Unreliable Communication
  - *Challenges*
    - Infinite State Space
    - Nondeterministic
  - *Goal: Evaluate Methods and Tools*

Theory ↓ *Code*

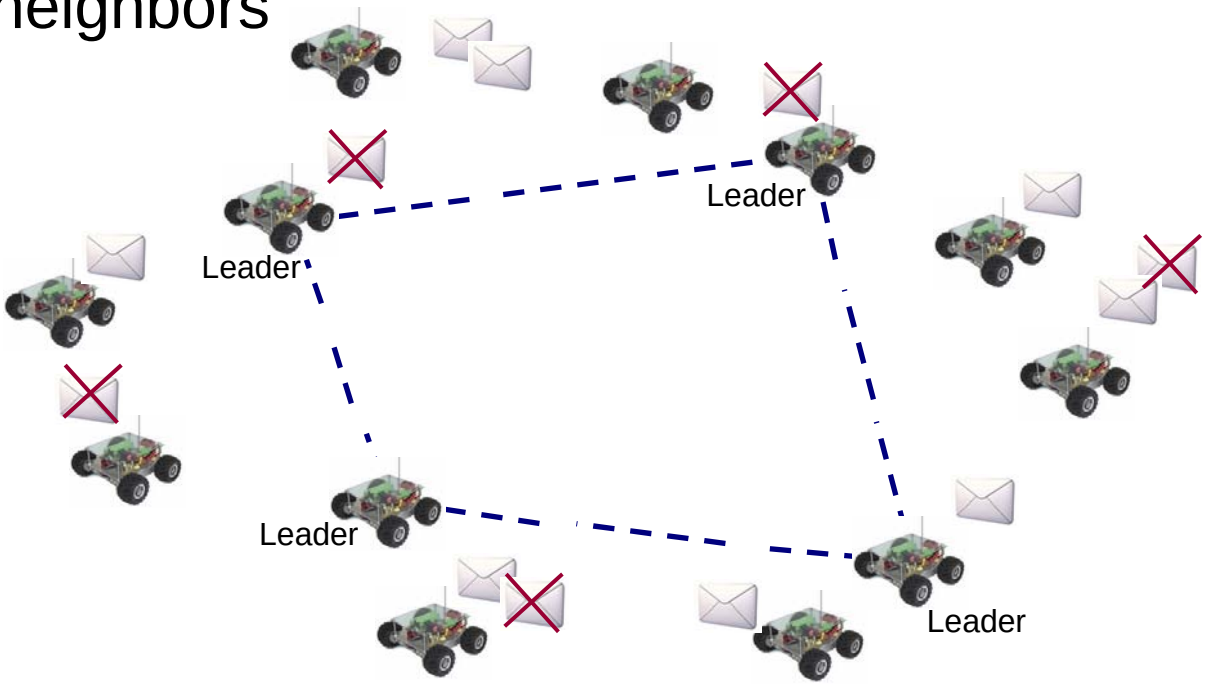
- Prove algorithms by hand
- Theorem Proving
- Model Checking
- Static Analysis
- Code walkthrough

Multi-Agent Systems (MAS)

- System consisting of collaborative agents
- Properties:
  - *No global coordination*
  - *Unreliable communication*
  - *Operating in adversarial environments*
  - *Discrete communication but continuous dynamics*
- Agents can be robots, sensors, software components
- Applications: DOD, homeland security, automatic surveillance, transportation…

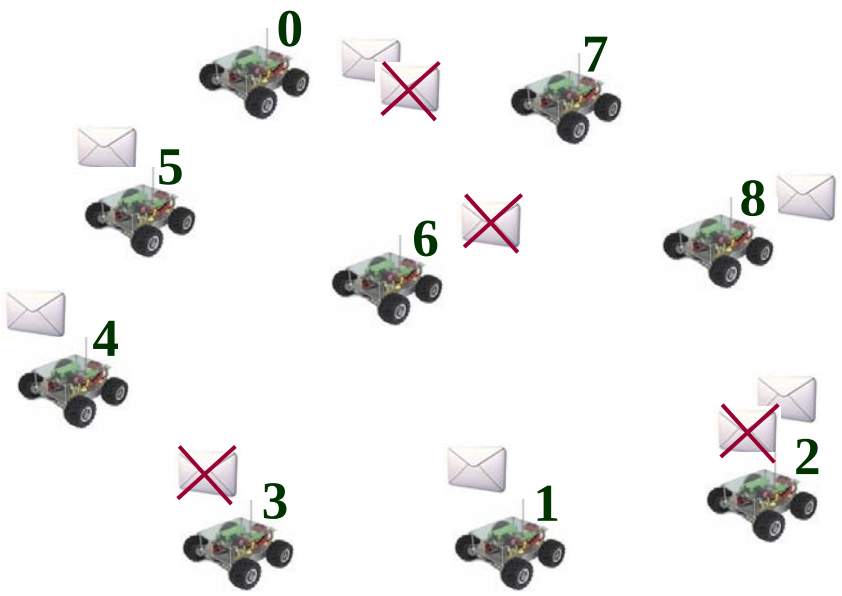
Example of MAS

- "*Robot Pattern Formation*" MAS
- MAS goal is "*forming and maintaining a formation while leaders move*"
- *Protocol of an Agent:* move towards the average of its left and right neighbors



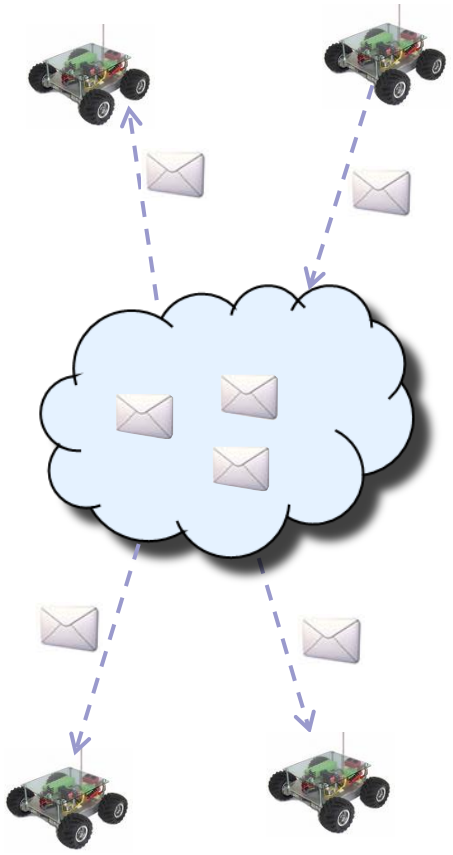
Example of MAS

- "*Robot Leader Election*" MAS
- MAS goal is "*agreeing on a leader*"
- *Protocol of an Agent:* set its value to the minimum between its current value and the value received



Unreliable Communication Medium

- *Unreliable Broadcast Communication Medium*:
  - Agents send and receive messages
  - Agent *i* broadcasts its value infinitely often
  - Messages in transit can be lost, delayed, copied, forged or received out-of-order



Challenges in Verifying MAS

- Fundamental Question:

  "Does the System behave according to the specification?"

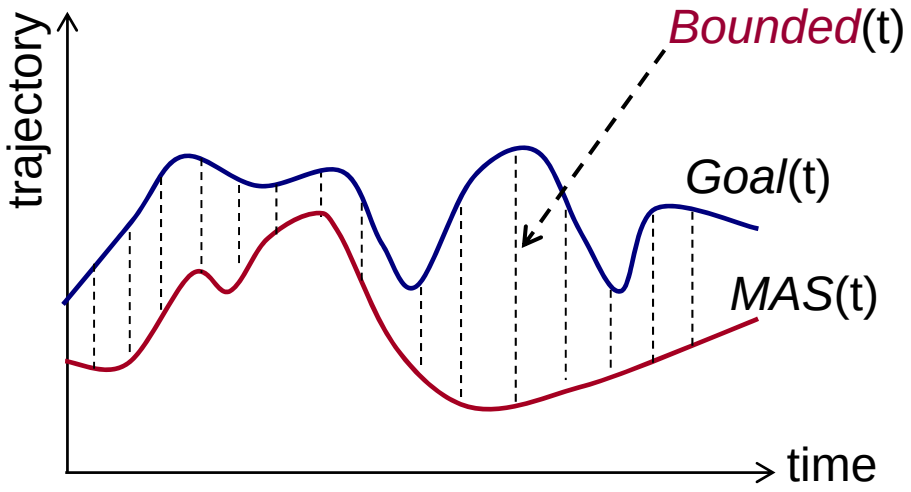
- Challenges: Reasoning about
  - Dense State Space
  - Unreliable Communication
- How?
  - *Theory*: Abstraction
  - *Executable Code*: Use Abstraction to build Reusable Components

Outline

- Reliable Multi-Agent Systems (MAS) with Unreliable Communication
  - *Challenges*
    - Infinite State Space
    - Highly nondeterministic
  - *Goal: Evaluate Tools and Methods*
    - Prove algorithms by hand
    - Theorem Proving
    - Model Checking
    - Static Analysis
    - Code walkthroughs

Procedure for MAS

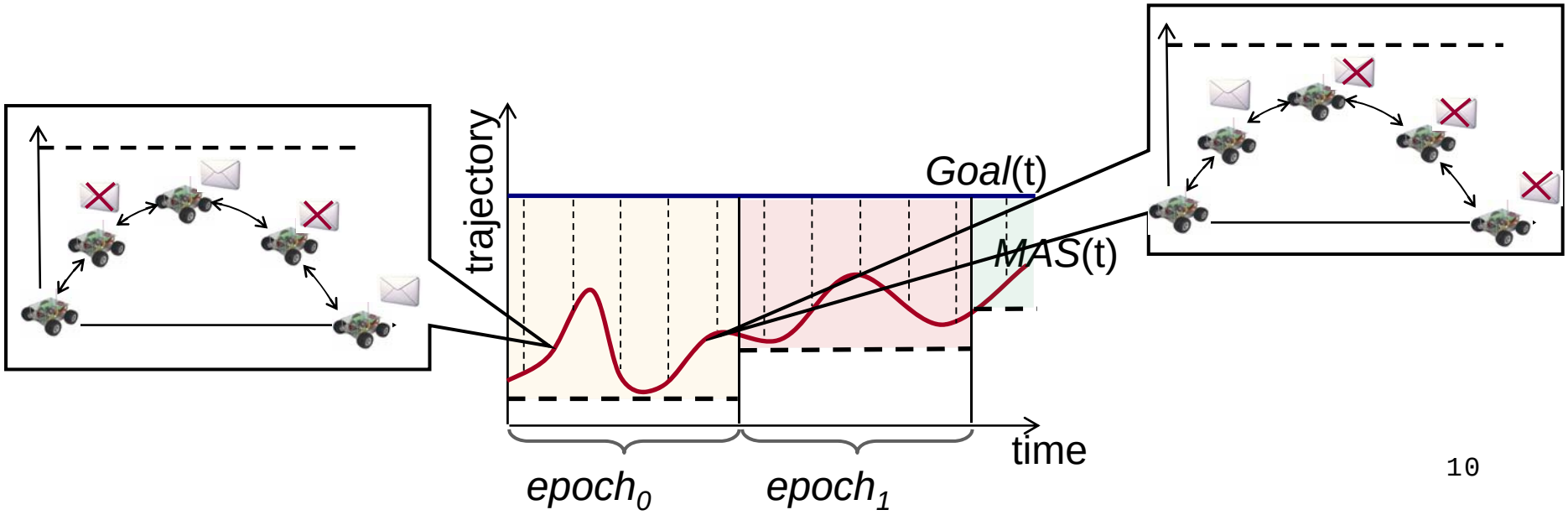
- Systematic procedure for designing reliable MAS:
  - *No concern about messages*
  - *No concern about dynamics*



- K.M. Chandy, B. Go, S. Mitra, C. Pilotto, and J. White. “Verification of distributed systems with local-global predicates”. *Formal Aspect of Computing*, 2010.
- K.M. Chandy, S. Mitra, and C. Pilotto. “Convergence verification: From shared memory to partially synchronous systems”. *FORMATS*, 2008.
- K.M. Chandy, C. Pilotto and J. White. “Consensus on Asynchronous Communication Networks in Presence of External Input”. Submitted, 2010.

Epochs

- Systematic procedure for designing reliable MAS.
- Conditions for
  - *Bounded Steady-State Error (Additive MAS)*
  - *Zero Steady-State Error*
- Epochs ensure decreasing error along any execution



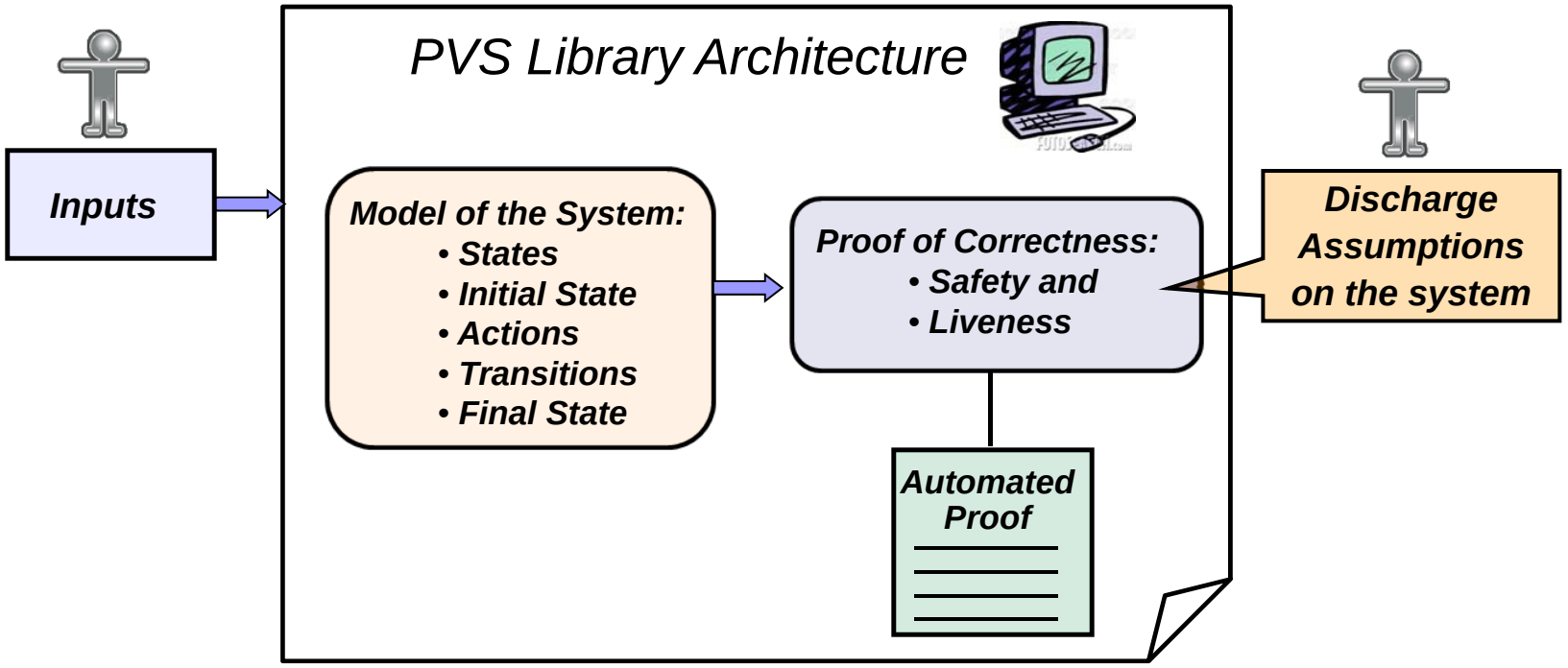
Outline

- Reliable Multi-Agent Systems (MAS) with Unreliable Communication
  - *Challenges*
    - Infinite State Space
    - Highly nondeterministic
  - *Goal: Evaluate Tools and Methods*
    - Prove algorithms by hand
    - Theorem Proving
    - Model Checking
    - Static Analysis
    - Code walkthrough

Theorem Proving Tool

- PVS (Prototype Verification System)
- Theorem proving is challenging
  - What should be proved?
  - Requires expertise
  - Lack of adequate libraries
- Our work
  - Builds reusable libraries
  - Minimizes proof obligations for end users
  - www.infospheres.caltech.edu

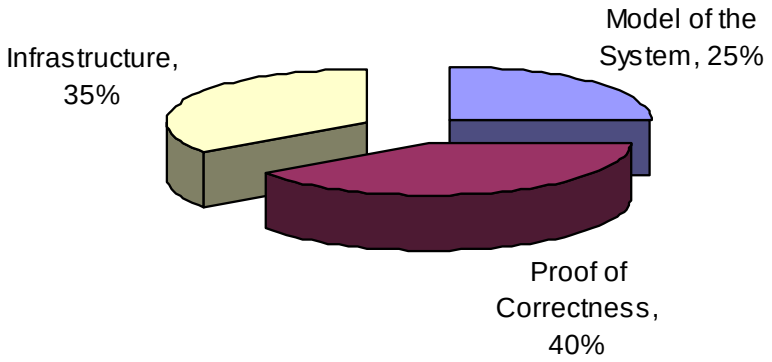
PVS Verification Framework



- C. Pilotto and J. White. “Verification of Faulty Message-Passing Systems with Continuous State Space”. Second NASA Formal Methods Symposium (NFM), 2010.

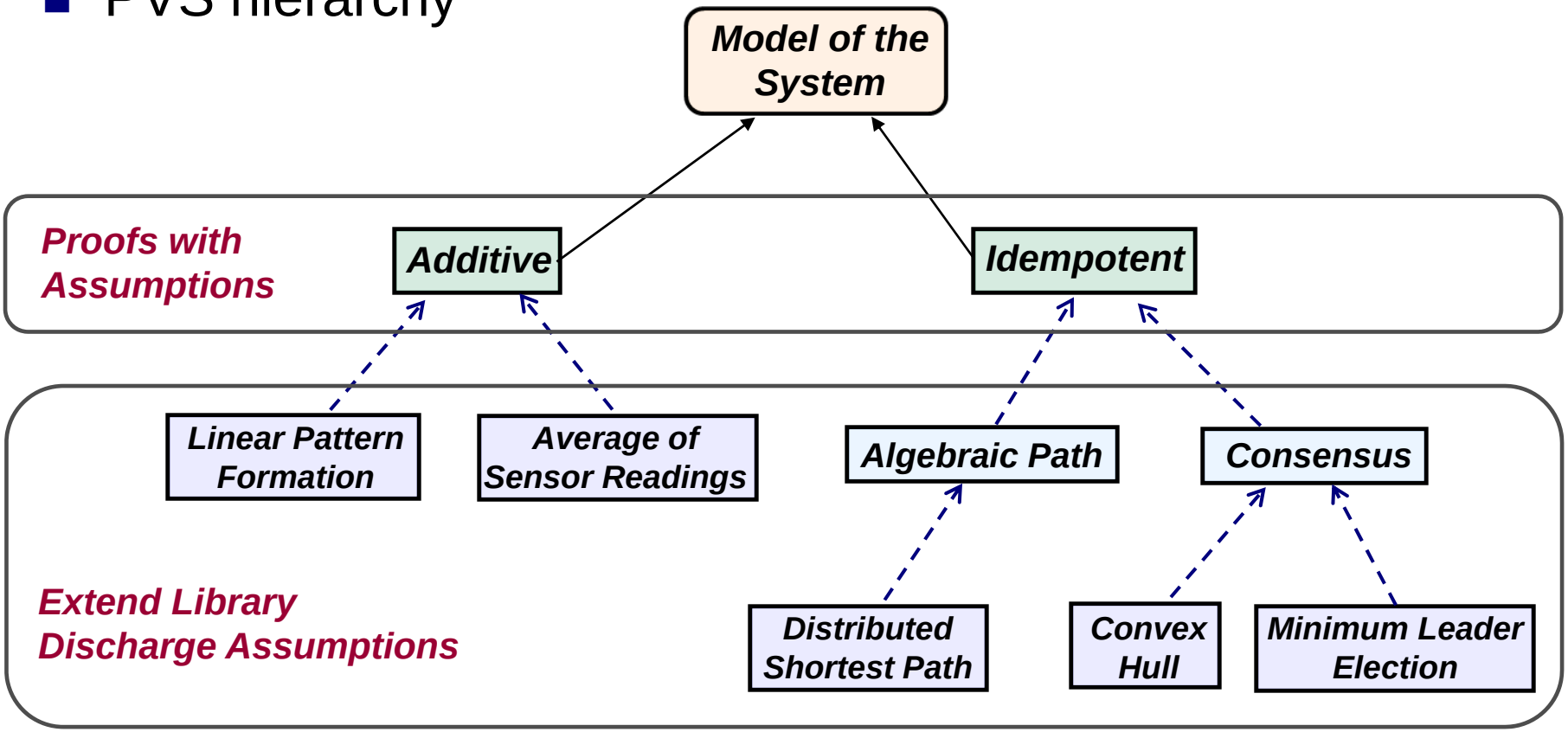
Discussion

- Extensive library
  - ~150 lemma
  - ~6000 proof steps
  - ~10 PVS theories
- Infrastructure extends NASA PVS libraries
- Applications:
  - Robot Pattern Formation: ~1500 proof steps
  - Robot Leader Election: ~120 proof steps



Discussion of Theorem Proving for MAS

- Verification of MAS with arbitrary number of agents
- PVS hierarchy



Leader Election in PVS

- PVS code:

```
MinLeaderElection: THEORY
BEGIN
    min (m, n: real): real = IF m > n THEN n ELSE m ENDIF
    IMPORTING consensus [posnat, real, min, posinf, >=]
END MinLeaderElection
```

- PVS TCC:

```
% Assuming TCC generated (at line 10, column 12) for
% consensus [posnat, real, min, posinf, >=]
IMP_consensus_TCC5: OBLIGATION FORALL (u: real): min (u, u)= u
```

Outline

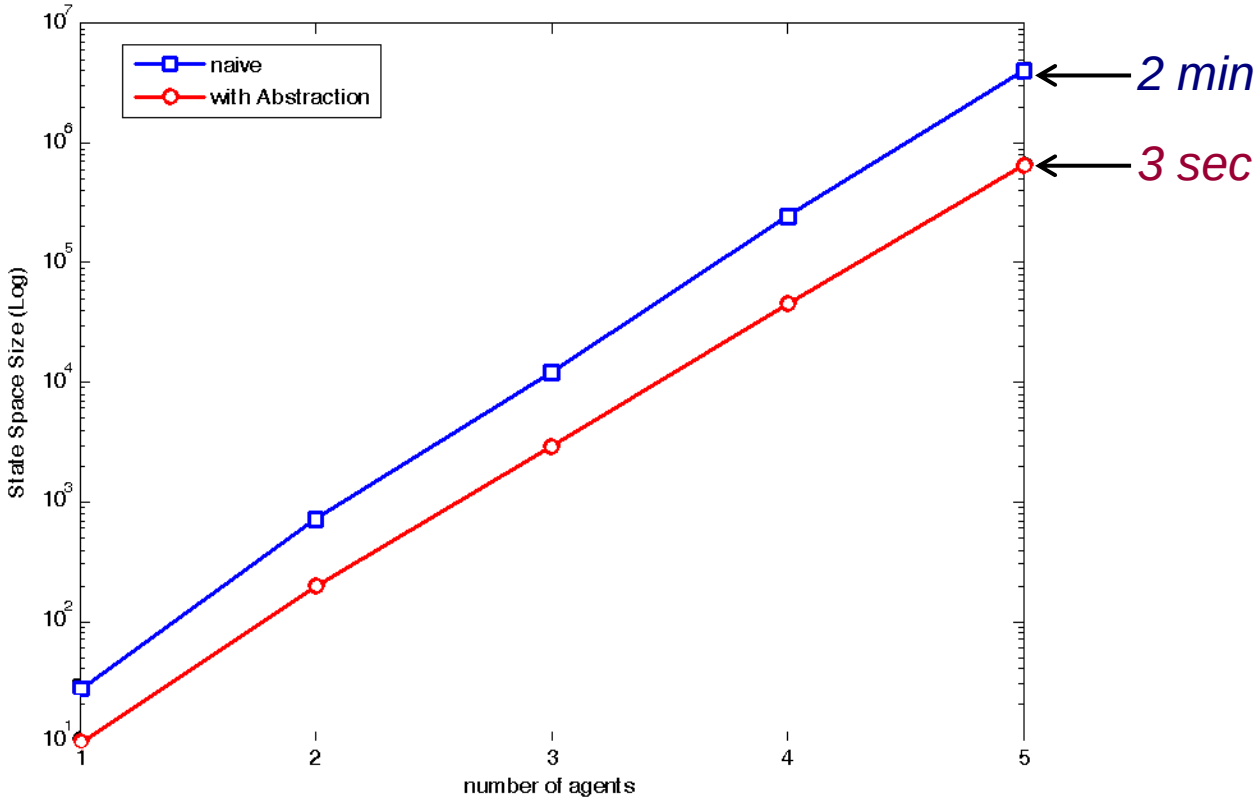
- Reliable Multi-Agent Systems (MAS) with Unreliable Communication
  - *Challenges*
    - Infinite State Space
    - Highly nondeterministic
  - *Goal: Evaluate Tools and Methods*
    - Prove algorithms by hand
    - Theorem Proving
    - Model Checking
    - Static Analysis
    - Code walkthroughs

Model Checking MAS

- Spin (Simple Promela Interpreter)
  - LTL, asynchronous processes
  - Promela model of the system
- Model checking is challenging
  - Identifying proper aspects to model
  - Dynamics?
  - Communication?
  - Dense state space?
  - Reusability?

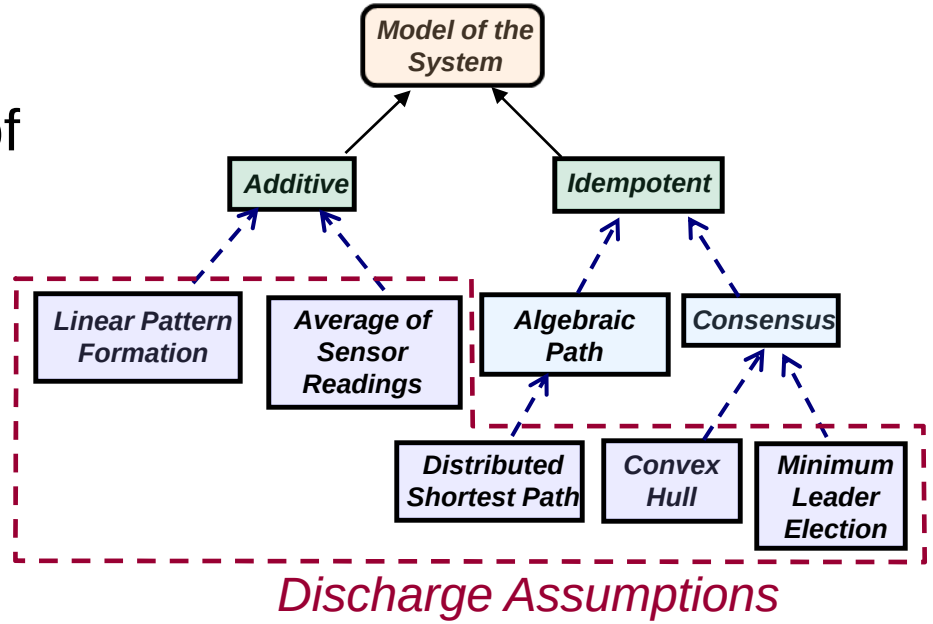
Model Checking Robot Leader Election

- Coding:
  - Naïve: 52 lines of code
  - with Abstraction: 43 lines of code



Discussion of Model Checking MAS

- Using Abstraction
  - Check only “assumptions”
  - Reduce state space: size of the buffer, dynamics
- Robot Pattern Formation: similar results
- Limitations
  - No reusability: no abstract operator
  - Discharge Assumptions at the bottom layer



Outline

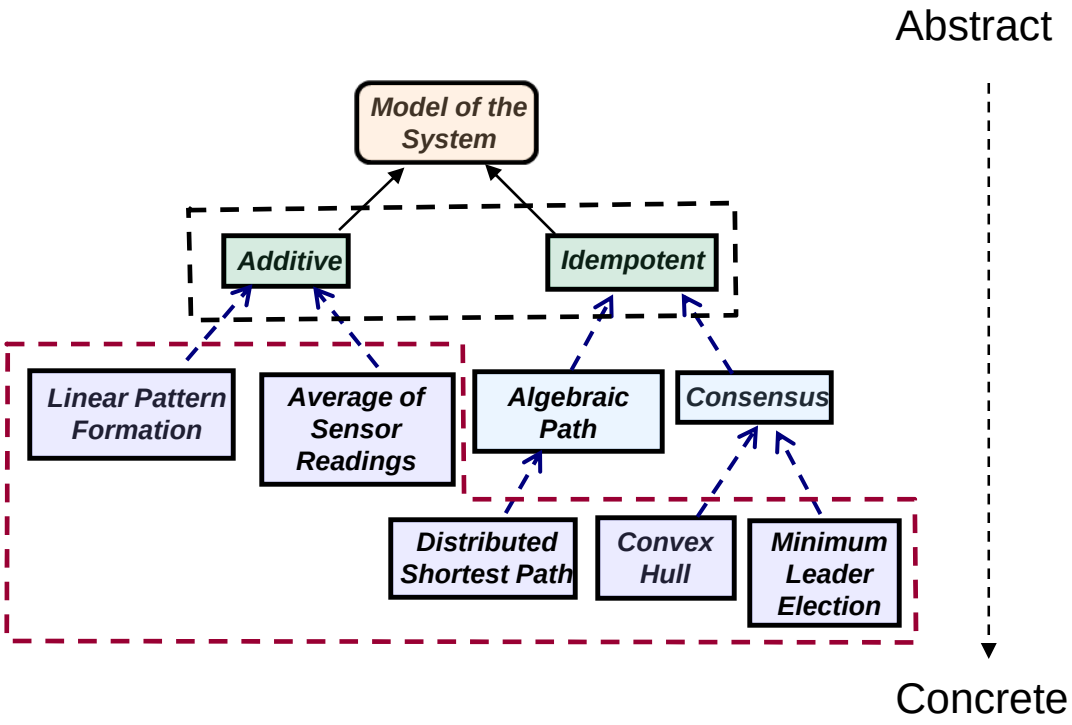
- Reliable Multi-Agent Systems (MAS) with Unreliable Communication
  - *Challenges*
    - Infinite State Space
    - Highly nondeterministic
  - *Goal: Evaluate Tools and Methods*
    - Prove algorithms by hand
    - Theorem Proving
    - Model Checking
    - Static Analysis
    - Code walkthroughs

Contract Specification

- Coding
  - JML (Java Modeling Language)
  - Spec#
- Specification
  - Annotate pre- and post-conditions
  - Reusability?
- Using Abstraction
  - Refinement of operators

JAVA with JML Annotations

- Library:
- Abstract Layers
  - Check Assumptions
- Concrete Layers
  - Derive checks from Previous Layers
  - Check operator correctness



Static Analysis of Robot Leader Election

- Discharged at the Leader Election Layer:

```
/*@
  @ also ensures(\result <= x && \result <= y &&
  @ (\result == x || \result == y));
  @*/
int min(int x, int y) {
    if (x < y) return x;
    else return y;
}
```

Conclusion

- *These methods and tools do help*
  - Theory
    - Prove algorithms by hand
    - Theorem Proving
    - Model Checking
    - Static Analysis
    - Code walkthrough
  - Code
- *to design reliable MAS with Unreliable Communication*
  - Infinite State Space
  - Nondeterministic
- Thanks to AFOSR-MURI