Formal Model–Driven Design of Distributed Algorithms

Morten Kühnrich

Distributed and Embedded Semantics,
Aalborg University, Denmark

November 15, MEMICS'08, Czechia

Problem

Distributed algorithms are used in

- the Internet, e.g. routing, leader election, agreement protocols
- Database systems
- Scientific Computing

Distributed algorithms are often hard to

- test
- verify
- prove correct

Problem statement

How may development of distributed algorithms be supported?

Concretely

1. As case study: Distributed Agreement
2. Chandra and Toueg, *Unreliable Failure Detectors for Reliable Distributed Systems*, 1996.
3. Model their algorithm
4. Improve algorithm iteratively
5. Verify possible improvements

The Agreement Problem

Agreement is the following problem:

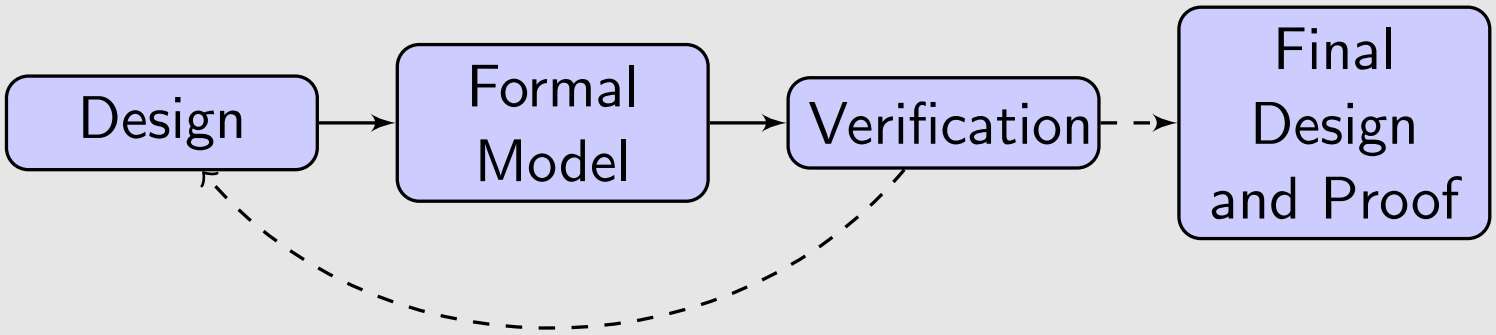
1. Each agent has a unique id i
2. n agents each propose a value $v_i \geq 0$.
3. at some time everyone must agree one value v_i.

To simplify matters we assume $v_i = i$ for all i.

We require that

1. every live agent eventually picks a value, i.e. **Termination**
2. every live agent decide on the same value, i.e. **Agreement**

Formal Model–Driven Design



- created by Uppsala University, Sweeden and Aalborg University, Denmark.
- an integrated tool for
- model checking of
- timed automata.

Related Work

Automatic generation of Distributed Algorithms

Piotr Zieliński. Automatic verification and discovery of Byzantine Consensus protocols. (DSN 2007)

Model driven development

Mellor, Clark, and Futagami. Guest editors' introduction: Model-driven development. IEEE Software, 2003.

Hand crafted development

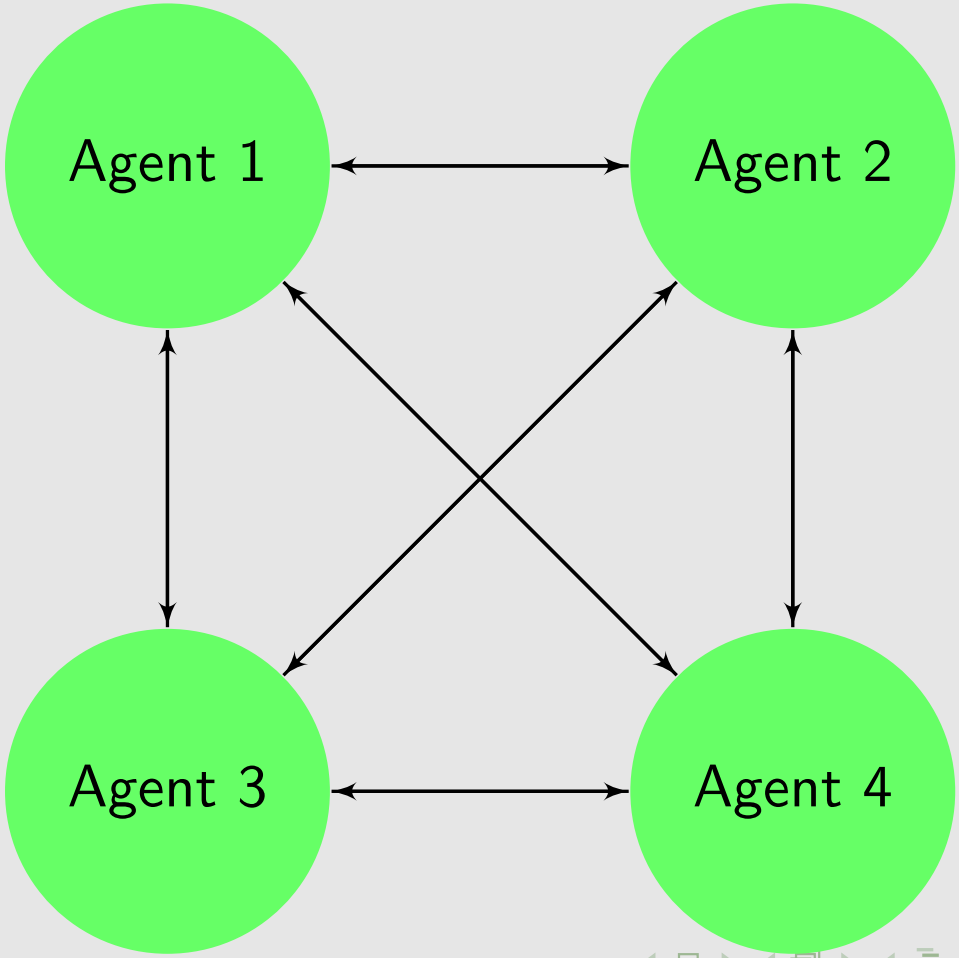
Nancy A. Lynch. Distributed Algorithms. San Francisco, 1996.

Model check of agreement algorithms

Tatsuhiro Tsuchiya and André Schiper. Using bounded model checking to verify consensus algorithms, 2008.

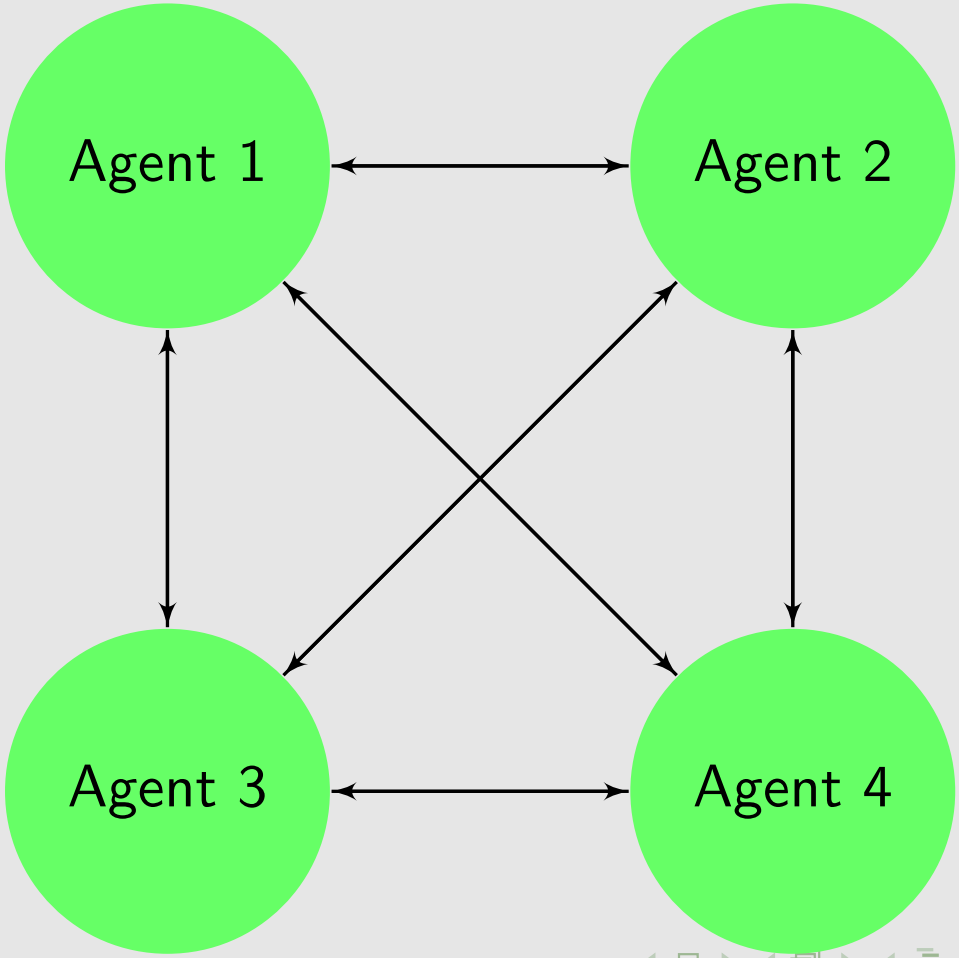
Starting point

1. Each agent has a unique id



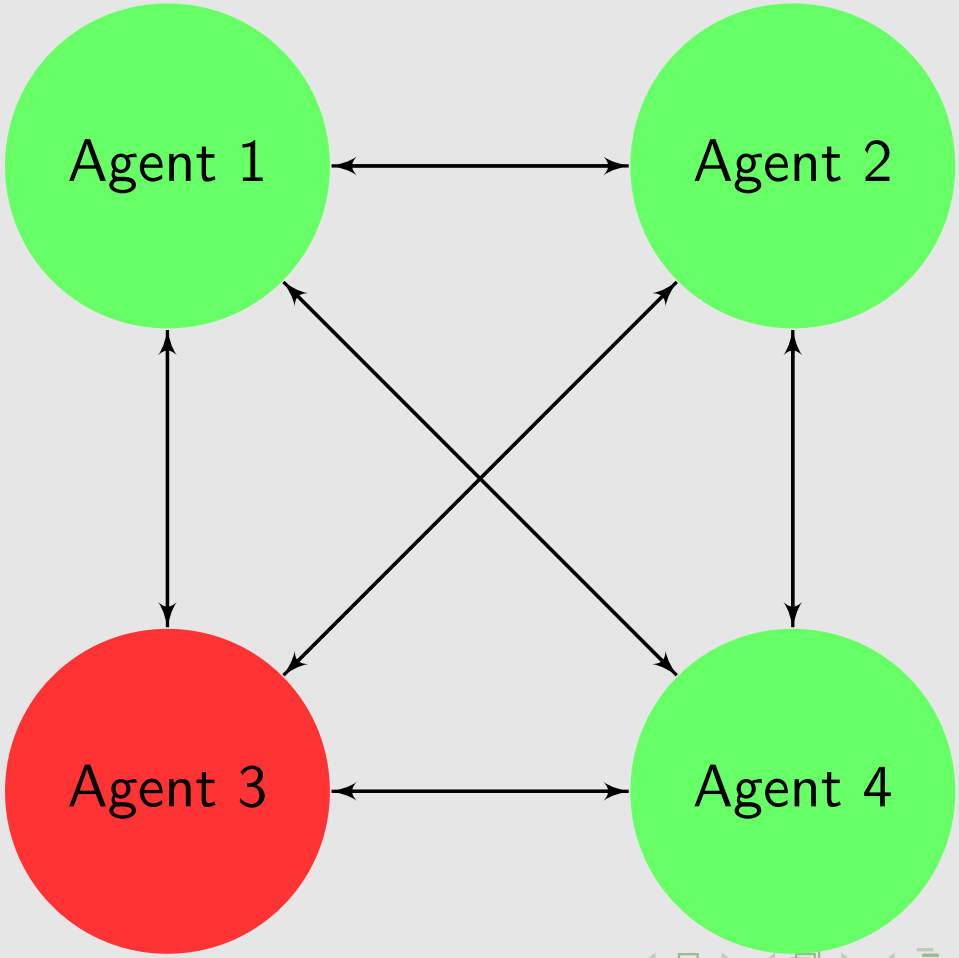
Starting point

1. Each agent has a unique id
2. Asynchronous communication



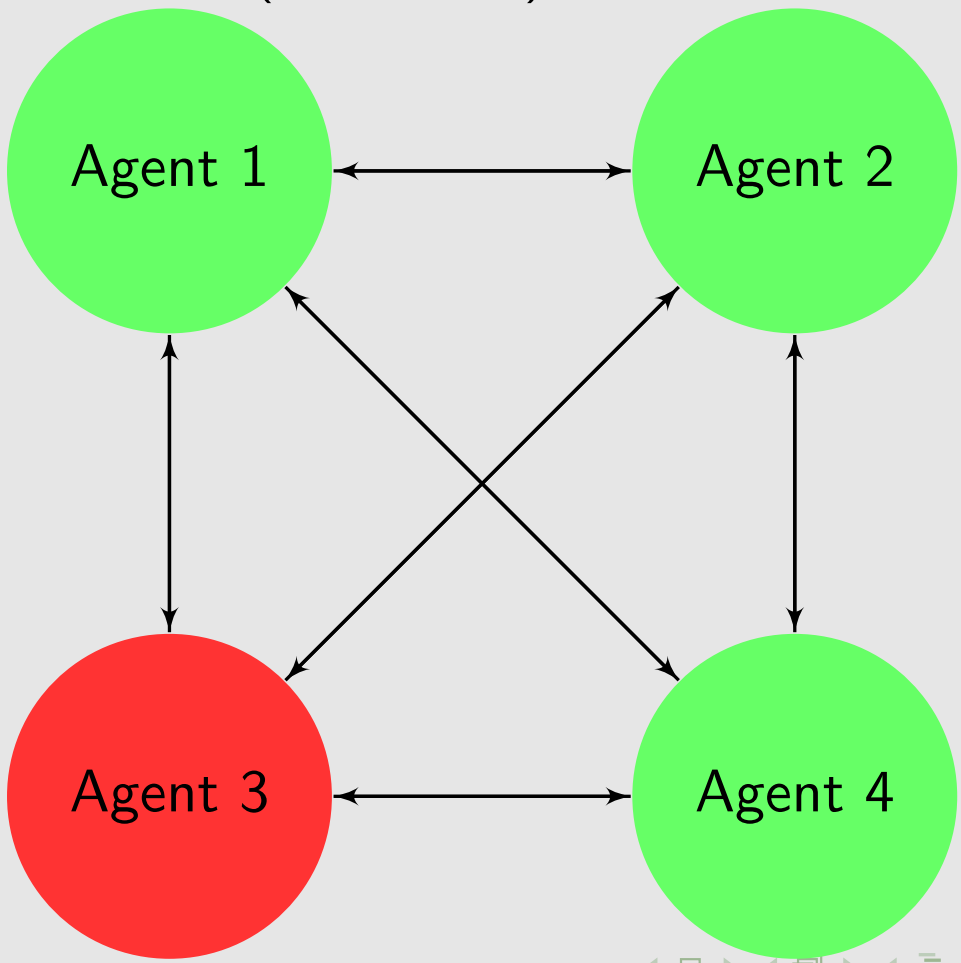
Starting point

1. Each agent has a unique id
2. Asynchronous communication
3. Process Failures



Starting point

1. Each agent has a unique id
2. Asynchronous communication
3. Process Failures
4. Failure Detectors (Unreliable)



Details on unreliable failure detectors

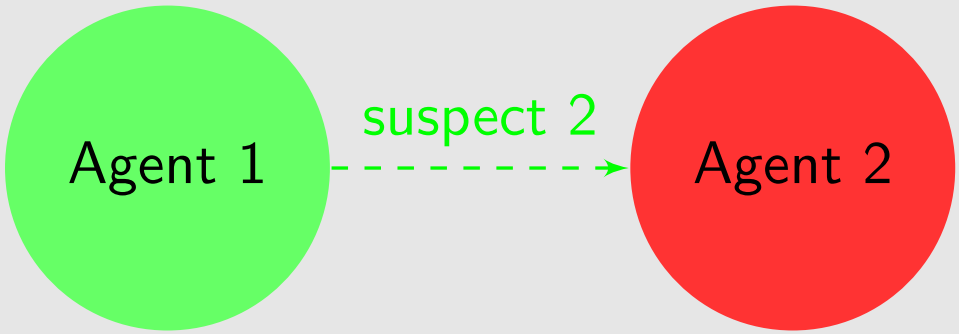
Detection of a failure

Details on unreliable failure detectors

Detection of a failure

Details on unreliable failure detectors

Detection of a failure

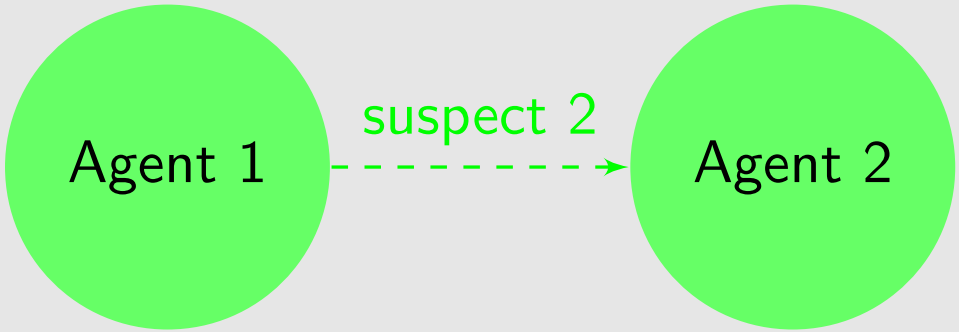


Details on unreliable failure detectors

Unreliable failure detectors may make mistakes!

Details on unreliable failure detectors

Unreliable failure detectors may make mistakes!



Details on unreliable failure detectors...

1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)

Details on unreliable failure detectors...

1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)

Details on unreliable failure detectors...

1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)

Details on unreliable failure detectors...

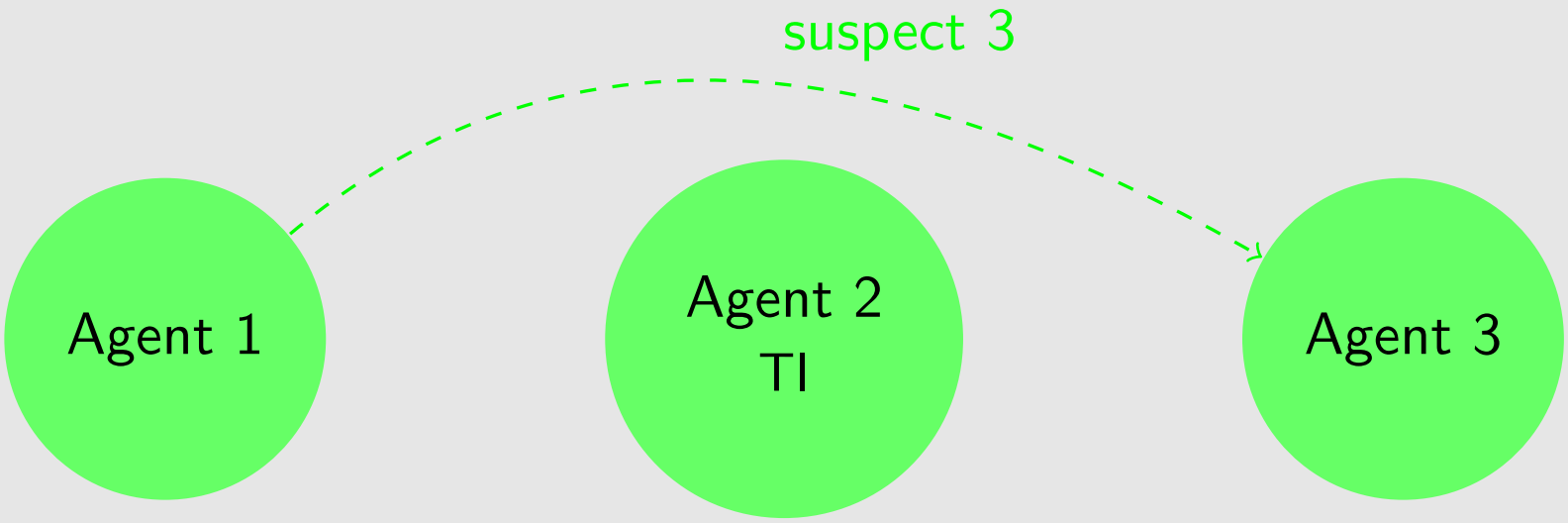
1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)

Details on unreliable failure detectors...

1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)

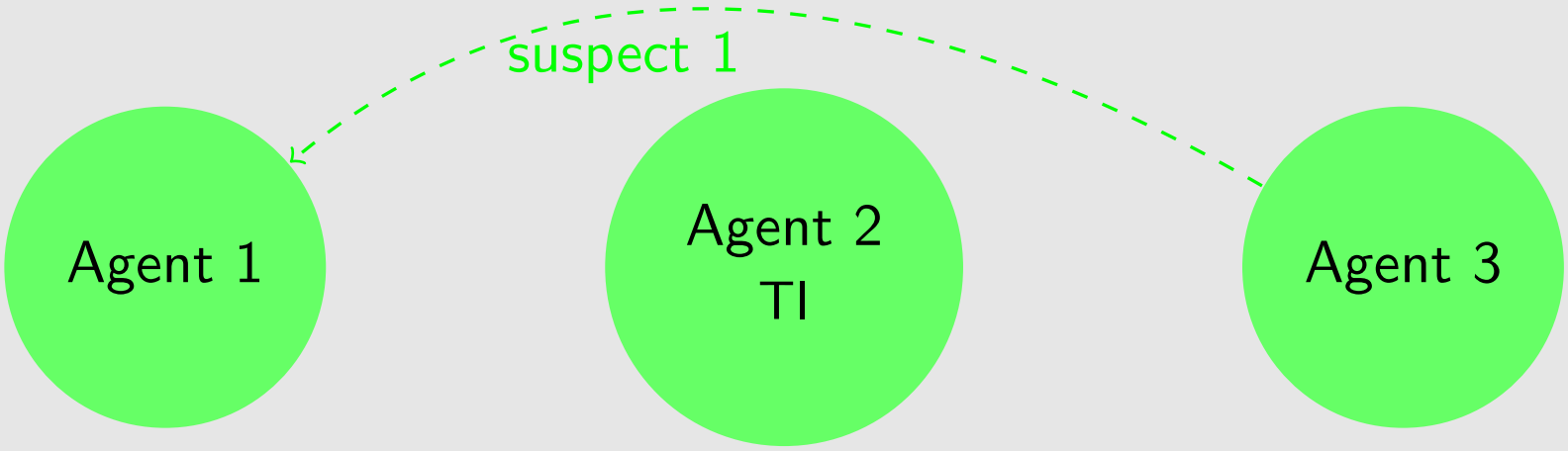
Details on unreliable failure detectors...

1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)



Details on unreliable failure detectors...

1. At least one live agent is never wrongly suspected.
2. For modeling purposes we choose one such agent – called the trusted immortal (in short TI)



Chandra & Toueg

Each agent j is equipped with

1. a knowledge vector V_j
2. a relay vector Δ_j

Initialization (for three agents)

$$V_1 = \Delta_1 = (1, \perp, \perp)$$
$$V_2 = \Delta_2 = (\perp, 2, \perp)$$
$$V_3 = \Delta_3 = (\perp, \perp, 3)$$

Chandra & Toueg

Three phases:

1. Learn values from others or suspect ($n-1$ rounds)
2. Learn $\perp$'s from others or suspect (1 round)
3. Decide on a value (first non-$\perp$ value)

Notice that agents are not synchronized!

Some may be in Phase 1 while others are in Phase 3 etc.

Theorem (Chandra & Toueg)

The algorithm satisfies the

1. *termination and*
2. *agreement property.*

When run on n agents it uses $n - 1$ rounds worst– and best case.

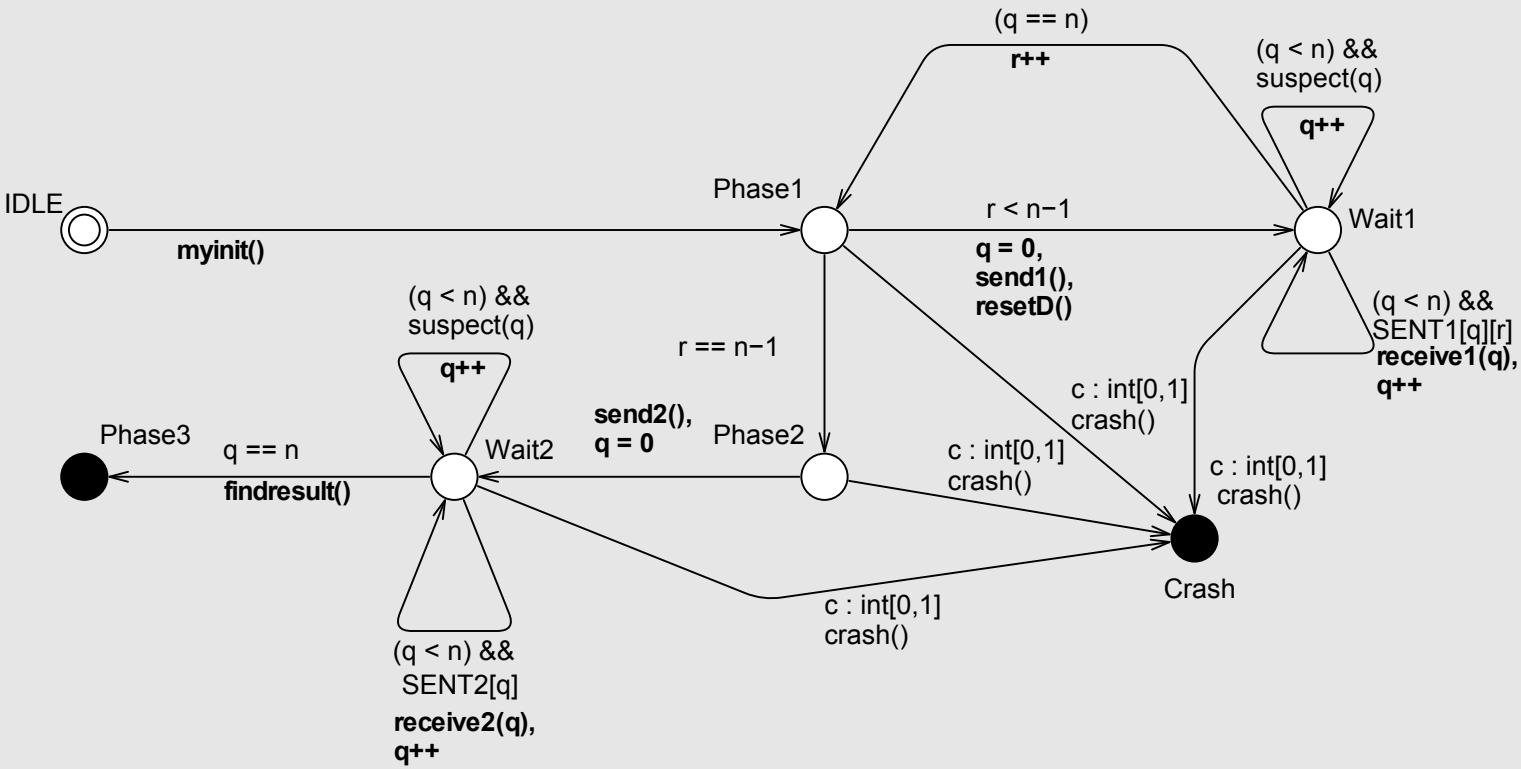
An example run

Agent 1	Agent 2(TI)	Agent 3
Phase 1	Phase1	Phase1
$(1, \perp, \perp)$	$(\perp, 2, \perp)$	$(\perp, \perp, 3)$
$1, \perp, \perp$	$\perp, 2, \perp$	$\perp, \perp, 3$
Round 1	Round 1	Round 1
Rcv 1,2	Rcv 2	Rcv 1,2,3
Susp 3	Susp 1,3	
$(1, 2, \perp)$	$(\perp, 2, \perp)$	$(1, 2, 3)$
$\perp, 2, \perp$	$\perp, \perp, \perp$	$1, 2, \perp$
Crash	Round 2	Round 2
	Rcv 2	Rcv 2,3
	Susp 1,3	Susp 1
	$(\perp, 2, \perp)$	$(1, 2, 3)$
	$\perp, \perp, \perp$	$\perp, \perp, \perp$

An example run...

Agent 1	Agent 2(TI)	Agent 3
	Phase 2	Phase 2
	$(\perp, 2, \perp)$	$(1, 2, 3)$
	Rcv 2	Rcv 2,3
	Susp 3	
	$(\perp, 2, \perp)$	$(\perp, 2, \perp)$
	Phase 3	Phase 3
	Decide 2	Decide 2

UPPAAL automaton



Verification

$$\texttt{A[]}\ \texttt{deadlock} \Rightarrow \quad (1)$$
$$\forall \texttt{i} : (\texttt{Agent(i).Crash or Agent(i).Phase3})$$

$$\texttt{A[]} \exists\ \texttt{y}\ \forall \texttt{i} : \quad (2)$$
$$(\texttt{Agent(i).Phase3} \Rightarrow \texttt{Agent(i).decide} == \texttt{y})$$

No problems! (278316 states explored, in 5.2 secs.)

Fast agreement example run

Agent 1	Agent 2(TI)	Agent 3
Phase 1	Phase1	Phase1
$(1, \perp, \perp)$	$(\perp, 2, \perp)$	$(\perp, \perp, 3)$
$1, \perp, \perp$	$\perp, 2, \perp$	$\perp, \perp, 3$
Round 1	Round 1	Round 1
Rcv 1,2	Rcv 1,2	Rcv 1,2,3
Susp 3	Susp 3	
$(1, 2, \perp)$	$(1, 2, \perp)$	$(1, 2, 3)$
$\perp, 2, \perp$	$1, \perp, \perp$	$1, 2, \perp$
Crash	Round 2	Round 2
	Rcv 2	Rcv 2,3
	Susp 1,3	Susp 1
	$(1, 2, \perp)$	$(1, 2, 3)$
	$\perp, \perp, \perp$	$\perp, \perp, \perp$

Fast agreement example run...

Agent 1	Agent 2(TI)	Agent 3
	Phase 2 $(1, 2, \perp)$ Rcv 2 Susp 3 $(1, 2, \perp)$	Phase 2 $(1, 2, 3)$ Rcv 2,3 $(1, 2, \perp)$
	Phase 3 Decide 1	Phase 3 Decide 1

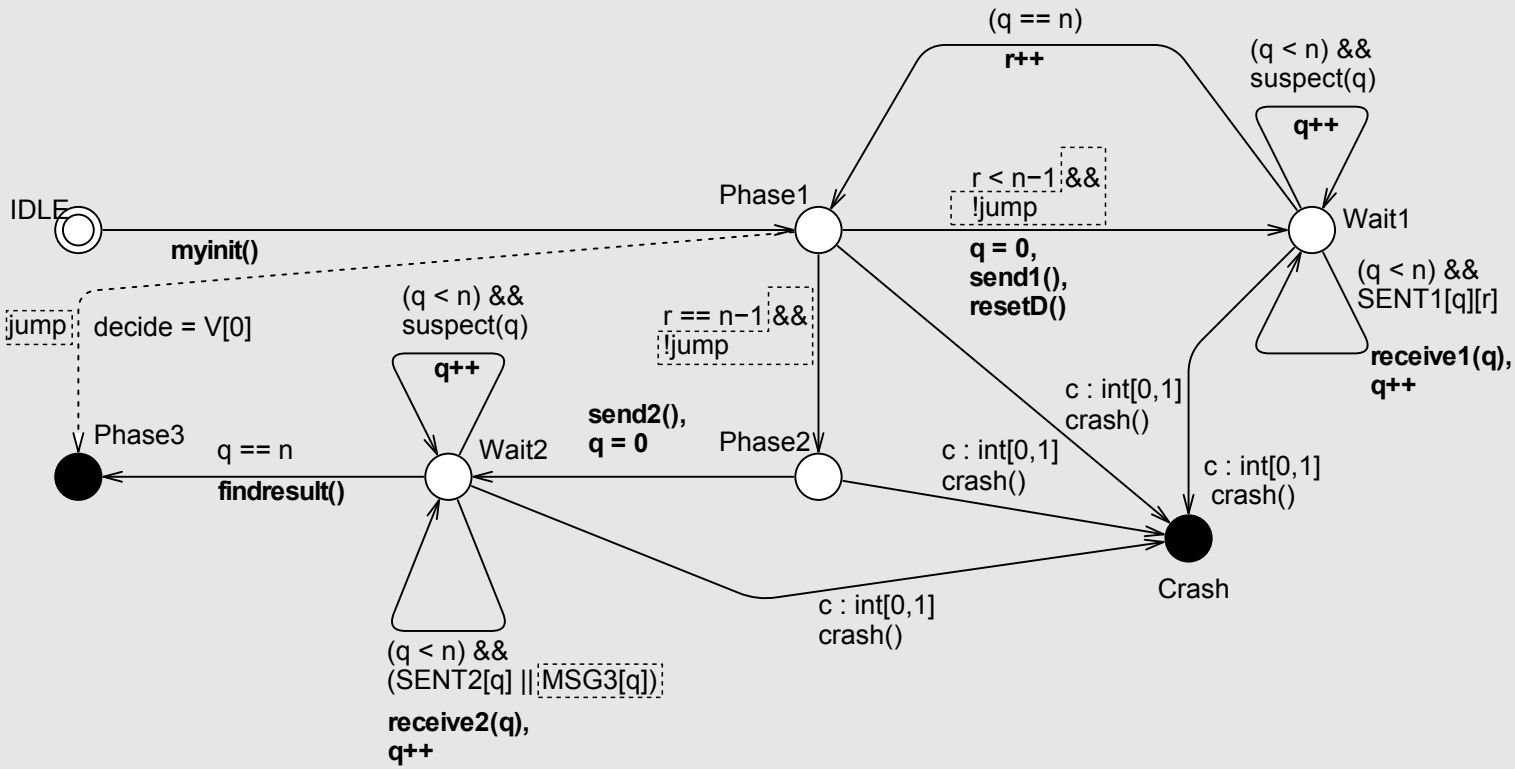
An Idea of An Improvement

When an agent discovers a match

1. broadcast a **stop**–message, and
2. jump to Phase 3

On reception of a **stop**–message jump to Phase 3.

An extended UPPAAL automaton



An error is found

The termination property breaks down for $n = 3$ agents:

$$\texttt{A[] deadlock} \Rightarrow$$
$$\forall \texttt{i} : (\texttt{Agent(i).Crash or Agent(i).Phase3})$$

154592 states explored, 4.7 secs. used.

Counter example

A deadlock may occur in the following way

1. Assume $\mathrm{TI} = 1$
2. Agent 2 discovers a match, broadcasts a stop, and jumps to Phase 3
3. Agent 1 receives this stop, jumps to Phase 3.
4. Agent 3 waits at Phase 2 for a message from 1.
5. It cannot suspect Agent 1...

Problem: Agent 1 jumped to Phase 3 without telling others!

Correction

When an agent discovers a match

1. broadcast a **stop**–message, and
2. jump to Phase 3

On reception of a **stop**–message, rebroadcast a stop-message and jump to Phase 3!

Verification and Main theorem

The model check of the improved algorithm for $n = 3$ agents is successful.

Theorem

The improved Algorithm satisfies the

1. *termination and*
2. *agreement property.*

When run on n–agents it uses

1. $n - 1$ *rounds worst–case and*
2. 2 *rounds in the bestcase.*

Conclusion

Formal model–driven development:

1. Models are easy to work with
2. Better intuition $\rightsquigarrow$ ideas/correctness arguments
3. Errors are found faster
4. Auto generated counter examples

Drawbacks:

1. The state space explosion problem
2. An algorithm for n agents may not work for $n + 1$ agents

Future work:

1. Apply formal model–driven development to other problems
2. Do qualitative studies of agreement algorithms:
   "How long time does it take to reach agreement?"