Verification of Faulty Message-Passing Systems with Continuous State Space in PVS

C. Pilotto and J. White

Computer Science Department
California Institute of Technology
Outline of the Talk

• Motivation
  – Robot Pattern Formations

• Systems of Linear Equations
  – Message-Passing Decentralized Scheme over Unreliable Communication

• PVS Framework
  – System Meta-theory
  – Proof of Correctness Meta-theory

• Conclusions and Future work
Pattern Formations

- Multi-agent System whose goal is forming “A fence around a given target”
Target Formation

• Multi-agent System whose goal is forming “A fence around a given target”

• In the final configuration, robots are equispaced
Protocol

• Multi-agent System goal is forming “A fence around a given target”

• In the final configuration, robots are equispaced
As System of Linear Equations

- Restrict on a single line
- Protocol corresponds to Gauss Iteration

\[
\begin{align*}
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-0.5 & 1 & -0.5 & 0 & 0 & 0 \\
\vdots \\
0 & 0 & 0 & -0.5 & 1 & -0.5 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_0 \\
x_1 \\
\vdots \\
x_{N-1} \\
x_N \\
\end{bmatrix}
= 
\begin{bmatrix}
\text{init}_0 \\
0 \\
\vdots \\
0 \\
\text{init}_N \\
\end{bmatrix}
\end{align*}
\]

\(x_0 = \text{init}_0\)

\(x_1 = 0.5 \times x_0 + 0.5 \times x_2\)

Average of its left and right neighbors

Stay put

Stay put
Outline of the Talk

• Motivation
  – Robot Pattern Formations

• Systems of Linear Equations
  – Message-Passing Decentralized Scheme over Unreliable Communication

• PVS Framework
  – System Meta-theory
  – Proof of Correctness Meta-theory

• Conclusions and Future work
Systems of Linear Equations

• **Goal**: Decentralized scheme for solving

\[ A \times x = b \]

• **Assumptions**:
  - \( A \) invertible with diagonal entries of 1

• **Decentralized System**: *Multi-Agent System* where agent \( i \) is responsible for computing \( x_i \)
Gauss Scheme

- Solving System of Linear Equations in Rounds starting from an initial guess init

\[ x_i(t) = \begin{cases} 
  b_i - \sum_{j \neq i} A(i, j) x_j(t-1) & t > 0 \\
  \text{init}_i & t = 0 
\end{cases} \]
Pattern Formation Challenges

- **Unreliable** Message-Passing Communication

- Agents do not update their positions *instantaneously*
Communication Medium

- **Unreliable Broadcast Communication Medium:**
  - Agents send and receive messages
  - Messages in transit can be lost, delayed or received out-of-order
  - Bounded transmission delay
  - Agent $i$ stores in the variable $msg_{ij}$ the last message it receives from agent $j$
  - Agent $i$ broadcasts $x_i$ infinitely often
Gauss over Unreliable Networks

• Over Message-Passing Systems:

\[ x_i(t) = \begin{cases} 
  b_i - \sum_{j \neq i} A(i, j) \text{msg}_{i,j} & \text{t}>0 \\
  \text{init}_i & \text{t}=0
\end{cases} \]
Agent Dynamics

- Agents do not update their positions *instantaneously*
- Introduce a variable $z_i$ storing its current position
- Variable $x_i$ stores its target position
- **Move action**: it moves from $z_i$ towards $x_i$ with some velocity for $dt$ time units

\[ z_i := 10 \]
\[ x_i := 8 \]
\[ z_i := 9 \]
Does this Scheme work?

If $A$ satisfies

(D1) invertible

(D2) diagonal entries equal to 1

$$\forall i : A(i,i) = 1$$

(D3) weakly diagonally dominant

$$\forall j : \sum_{k \neq j} |A(j,k)| \leq 1$$

(D4) strictly diagonally dominant in at least one row

$$\exists j : \sum_{k \neq j} |A(j,k)| < 1$$

then *Message-Passing Gauss Iteration* converges to $A^{-1} \cdot b$
Proof Summary

(E1) Error of the system $E$ does not increase

(E2) $E$ eventually decreases by a factor $\alpha$
Proof Summary

(E1) Error of the system $E$ does not increase:
   – For all actions of the system, the execution of the action does not increase the error

$E \leq k_1$
$E \leq k_2$
$E \leq k_3$

$k_1 > k_2 > k_3 \ldots$
(E2) E eventually decreases by a factor $\alpha$

• Proved by *Induction*

• Tree based on communication links in $A$
  – (i, j) is an edge if $A(i, j) \neq 0$
  – Roots: Strictly Diagonally Dominant agents

• Induction Proof: Assume $E = C$
  – *Base Case*: Error of Roots eventually $\alpha C$
  – *Induction Step*: assuming error along path to agent j (exclusive of j) is $\alpha C$ then eventually error of j is $\alpha C$
Outline of the Talk

• Motivation
  – Robot Pattern Formations

• Systems of Linear Equations
  – Message-Passing Decentralized Scheme over Unreliable Communication

• PVS Framework
  – System Meta-theory
  – Proof of Correctness Meta-theory

• Conclusions and Future work
PVS Verification Framework

Model of the System:
- States
- Initial State
- Actions
- Transitions
- Final State

Proof of Convergence:
- System Error
- Stability
- Progress

Interactive Proof: D1-D4

Mathematical Theories
- Vector
- Tree
- Matrix

System Theory
- System
  state, initial state predicate, actions, transition function, enabled predicate

Verification Theories
- Error Model
- Proof of Convergence
Description of the System

• **Automaton** with:
  – States
  – Initial State Predicate
  – Actions
  – Transition Function
  – Enabling Predicate

• Extend **time_machine metatheory**

```plaintext
IMPORTING time_machine[S, State
  ACS, Actions
  enabled, Enabling Predicate
  trans, Transition function
  start?, ...] Initial State Predicate
```
State of the System

- State of the System is given by the composition of State of Agents and State of the Communication Medium

\[
S: \text{TYPE} = [#
\begin{align*}
\text{target} & : \text{Vector, State of Agent} \\
\text{lastmsg} & : \text{Matrix, State of Agent} \\
\text{buffer} & : [\text{Index, Index} \rightarrow \text{Pset}] \\
\text{now} & : \text{nonneg\_real, Global Clock} \\
\text{next} & : [\text{Index} \rightarrow \text{nonneg\_real}] 
\end{align*}
#]
\]
State of the Agents

Index : TYPE = below(N)  Agents’ Identifiers
Vector: TYPE = [Index -> real]
Matrix: TYPE = [Index, Index -> real]

S: TYPE = [#
  target : Vector,
  lastmsg : Matrix,
  buffer : [Index, Index -> Pset],
  now : nonneg_real,
  next: [ Index -> nonneg_real]
#]
State of Communication Medium

\[
S: \text{TYPE} = [\#
\begin{align*}
\text{target} : & \text{Vector}, \\
\text{lastmsg} : & \text{Matrix}, \\
\text{buffer} : & [\text{Index, Index} \rightarrow \text{Pset}], \\
\text{now} : & \text{nonneg\_real}, \\
\text{next} : & [\text{Index} \rightarrow \text{nonneg\_real}]; \\
\#
\end{align*}
\]

Communication Medium – Buffer(i,j) dedicated channel from i to j

Next time a Send action is executed – Model infinitely often

---

Current value  Sender of the Message

\[
\begin{align*}
\text{Msg:} & \quad \text{TYPE} = [\# \text{loc: real, id: Index} \#] \\
\text{Pkt:} & \quad \text{TYPE} = [\# \text{msg: Msg, ddl: posreal} \#] \\
\text{Pset:} & \quad \text{TYPE} = \text{set[Pkt]} \\
\text{b:} & \quad \text{posreal} \\
\text{d:} & \quad \text{posreal}
\end{align*}
\]

Deadline – model bounded delay

---

Current value  Sender of the Message

\[
\begin{align*}
\text{Msg:} & \quad \text{TYPE} = [\# \text{loc: real, id: Index} \#] \\
\text{Pkt:} & \quad \text{TYPE} = [\# \text{msg: Msg, ddl: posreal} \#] \\
\text{Pset:} & \quad \text{TYPE} = \text{set[Pkt]} \\
\text{b:} & \quad \text{posreal} \\
\text{d:} & \quad \text{posreal}
\end{align*}
\]

Maximum delay

A channel is a set – model out-of-order messages

Upper bound on consecutive send actions executed by the same agent – Model infinitely often
Initial State Predicate

• Describe properties of the initial state

\[
\text{start?(s: S): bool =} \\
\text{now(s) = 0 \textit{Global clock 0}} \\
\text{AND} \\
\text{target(s)=x0 \textit{x initialized with init}} \\
\text{AND} \\
(\text{FORALL (i:Index): next(s)(i)\leq d}) \\
\text{AND} \\
(\text{FORALL (i,j: Index): lastmsg(s)(i,j)= x0(j)}) \\
\]

• Note that the communication channels can have messages in transit initially
Actions of the Systems

• The system has the following actions

ACS : DATATYPE BEGIN

nu_traj(delta_t:posreal): nu_traj?\textit{Advance Time –Time Manager}
send(p:Pkt, i:Index, d1:posreal): send?\textit{Agents}
receive(p:Pkt, i:Index): receive?\textit{Agents}
msgloss(p:Pkt, i:Index): msgloss?\textit{Packet loss - Channel}
move(i:Index, delta_t:posreal): move?\textit{Agents}
END ACS
Transition Function and Enabling Conditions

- **Body of the Actions**

  \[
  \text{trans} (a:ACS, s:S) : S = \\
  \text{CASES } a \text{ OF} \\
  \quad \text{nu\_traj} (\text{delta\_t}) : \ldots \\
  \quad \text{send} (p,i,d1) : \ldots \\
  \quad \text{receive} (p,i) : \ldots \\
  \quad \text{msgloss} (p,i) : \ldots \\
  \quad \text{move} (i,\text{delta\_t}) : \ldots \\
  \text{ENDCASES}
  \]

- **Enabling Conditions**

  \[
  \text{enabled} (a:ACS, s:S) : \text{bool} = \\
  \text{CASES } a \text{ OF} \\
  \quad \text{nu\_traj} (\text{delta\_t}) : \ldots \\
  \quad \text{send} (p,i,d1) : \ldots \\
  \quad \text{receive} (p,i) : \ldots \\
  \quad \text{msgloss} (p,i) : \ldots \\
  \quad \text{move} (i,\text{delta\_t}) : \ldots \\
  \text{ENDCASES}
  \]
Time Action

- Action

\[
\text{trans } (a:\text{ACS}, s:S): S =
\]
\[
\text{CASES } a \text{ OF}
\]
\[
\text{nu_traj } (\text{delta_t}): s \text{ WITH } [\text{now} := \text{now}(s) + \text{delta_t}]
\]
\[
\text{ENDCASES}
\]

- Enabling Conditions

\[
\text{enabled } (a:\text{ACS}, s:S): \text{bool} =
\]
\[
\text{CASES } a \text{ OF}
\]
\[
\text{nu_traj } (\text{delta_t}): \text{WITH}
\]
\[
\text{FORALL } (p: \text{Pkt}): \text{ddl}(p) >= \text{now}(s) + \text{delta_t}
\]
\[
\text{ENDCASES}
\]
Agent Send Action

• Agent i sends packet p

```
send (p: Pkt, i: Index, d1: posreal):
s  WITH [
    buffer := LAMBDA (k,j: Index ):
    IF ((k=i) AND (j /= i))
    THEN union (p, buffer(s)(k,j))
    ELSE buffer (s)(k,j)
    ENDIF,
    next := next(s) WITH [(i) := next(s)(i) + d1 ]
]
```

• Enabling Conditions

```
send (p, i, d1): next(s)(i) = now(s) Agent is allowed to send
    AND d1<=d Next scheduled time does not violate d
    AND id(msg(p))=i Send its current value and id
    AND loc(msg(p)) = target(s)(i)
    AND ddl(p) = now(s) + b Packet deadline does not violate b
```
Agent Receive Action

- Agent i receives packet p

\[
\text{receive} \ (p: \text{Pkt}, \ i: \text{Index}) : \\
\text{LET} \ m: \text{Msg} = \text{msg}(p), \\
\hspace{1cm} \text{sender and location of the message} \\
\hspace{1cm} j: \text{Index} = \text{id}(m), \\
\hspace{1cm} l: \text{real} = \text{loc}(m), \\
\hspace{1cm} \text{Ci: Vector} = \text{update}(\text{row(lastmsg}(s),i),j,l) \\
\text{IN} s \text{ WITH} [ \\
\hspace{1cm} \text{buffer := buffer}(s) \text{ WITH} \\
\hspace{2cm} [(j,i) := \text{remove}(p,\text{buffer}(s)(j,i))], \\
\hspace{1cm} \text{lastmsg := lastmsg}(s) \text{ WITH} [ (i,j):= l ], \\
\hspace{1cm} \text{target := target}(s) \text{ WITH} [ (i):= \text{gauss}(\text{Ci},i)]] \\
\]

- Enabling Conditions

\[
\text{receive} \ (p,i): \ p \text{ is in the channel from } j \text{ to } i \text{ and its deadline is not violated} \\
\hspace{1cm} \text{buffer}(s)(\text{id}(\text{msg}(p)), i)(p) \text{ AND } \text{ddl}(p) \geq \text{now}(s) \\
\]
Message Loss Action

• **Packet p is removed from the System**

```plaintext
trans (a:ACS, s:S): S =
CASES a OF
msgloss (p: Pkt, i: Index):
  LET
    m: Msg = msg (p),
    j: Index = id(m) Sender of the message
  IN s
  WITH [ buffer := buffer (s)
    WITH [ (j,i) := remove(p, buffer(s)(j,i)) ]]
ENDCASES
```

• **Enabling Conditions**

```plaintext
enabled (a:ACS, s:S): bool =
CASES a OF
msgloss (p, i): buffer (s)(id(msg(p)),i)(p)
ENDCASES
```

Sender of the message
Remove p from the channel
p is in the buffer from sender of p to i
Agent Move Action

- Agent $i$ moves from its current to its target value

```plaintext
move (i: Index, delta_t: posreal):
s WITH [
    lastmsg := lastmsg(s) WITH [(i,i) := target(s)(i)],
    now := now(s) + delta_t  \text{Advance Clock of delta}_t \text{ unit}
]
```

- Enabling Conditions

```plaintext
move (i: Index, delta_t: posreal):
    FORALL (p: Pkt): ddl (p) >= now (s) + delta_t \text{No packet deadline is violated}
```
Outline of the Talk

• Systems of Linear Equations
  – Message-Passing Decentralized Scheme over Unreliable Communication

• PVS Framework
  – System Meta-theory
  – Proof of Correctness Meta-theory

• Application
  – Robot Pattern Formations

• Conclusions and Future work
Assumptions

• Assumptions

ASSUMING

inverse_exist: ASSUMPTION inv?(A) \( D1 \)
diag_entry : ASSUMPTION FORALL (i:Index): A(i,i)=1 \( D2 \)
diag_dominant : ASSUMPTION dd?(A) \( D3 \)
strictly_diag_dominant : ASSUMPTION sdd?(A) \( D4 \)

ENDASSUMING

• where

\[
\text{dd?}(m): bool = \\
\text{FORALL} \ (r: \text{Index}): \text{sum} (\text{row} (\text{abs} (m),r), r) \leq \text{abs} (m(r,r))
\]

\[
\text{sdd?}(m): bool = \\
\text{EXISTS} \ (r: \text{Index}): \text{sum} (\text{row} (\text{abs} (m),r), r) < \text{abs} (m(r,r))
\]
Error Model

• Error of the system as the maximum of agent errors

\[
\text{me} : [S \rightarrow \text{nonneg\_real}]
\]

\[
\text{me\_all\_error}: \text{AXIOM FORALL}(i) : \text{mes}(s,i) \leq \text{me}(s)
\]

\[
\text{me\_ex\_error}: \text{AXIOM EXISTS}(i) : \text{mes}(s,i) = \text{me}(s)
\]

• Error of agent i in the system

\[
\text{mes}(s,i) : \text{nonneg\_real} = \max(\text{mae}(s,i), \text{dbe}(s,i))
\]

Error of i is the maximum of its error along its outgoing channels and its error in all agents
Proof Summary

(E1) Error of the system $E$ does not increase

\[\text{not_incr_error :} \]
\[\text{LEMMA enabled}(a,s) \implies \text{me}(s) \geq \text{me}(\text{trans}(a,s))\]

- Proof is a case on the actions

\[E \leq k_1\]
\[E \leq k_2\]
\[E \leq k_3\]

\[k_1 > k_2 > k_3 \ldots\]
Proof Summary

(E2) $E$ eventually decreases by a factor $\alpha$

- $p$ defined recursively along the tree $t$

$$p(i:\text{Index}) \text{ RECURSIVE nonneg\_real} =$$

$$\begin{align*}
\text{IF} & \quad \text{root}\_t?(t,i) \\
\text{THEN} & \quad \text{sum(row(abs(A),i),N-1,i)} \\
\text{ELSE} & \quad \text{abs(A}(i,\text{parent}(t,i))) \times p(\text{parent}(t,i)) + \\
& \quad \text{sum(row(abs(A),i),N-1,i,\text{parent}(t,i))}
\end{align*}$$

- showed that $0 \leq p < 1$

- $\alpha$ defined as the maximum of function $p$
Proof Summary

(E2) E eventually decreases by $\alpha$

- Define a family of predicates $Z$
- Prove Stability

$$Z_{\text{stable}}: \text{LEMMA } Z(s,C,i) \text{ AND enabled}(a,s) \text{ IMPLIES } Z(\text{trans}(a,s),C,i)$$

Proof is a case on the actions

- Prove Progress

Proof broken in smaller lemmas
Using the PVS Framework

PVS Library Architecture

Model of the System:
- States
- Initial State
- Actions
- Transitions
- Final State

Proof of Convergence:
- System Error
- Stability
- Progress

Interactive Proof: D1-D4

A, b, init

\[
A = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-0.5 & 1 & -0.5 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.5 & 1 & -0.5 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
b = \begin{pmatrix}
\text{init}_0 \\
0 \\
\vdots \\
0 \\
\text{init}_n
\end{pmatrix}
\]

Average of its left and right neighbors

Stay put

Stay put
Conclusions and Future Directions

• Extend results and framework to a richer class:
  – Non linear schemes
  – How much can we reuse?

• Focus on communication
  – Conditions for proving correctness of a message-passing scheme
  – Synchronous version of the scheme
  – How much can we reuse?
Framework for Reliability

- Initial State
- Algorithm
- Final State

Certificate of Correctness

Family of Predicates in Conjunctive Form

- Interactive Proof
  - Certificate of Correctness satisfies Assumptions