Computer Science at CALTECH
DIVISION OF ENGINEERING AND APPLIED SCIENCE

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

C. Pilotto and J. White

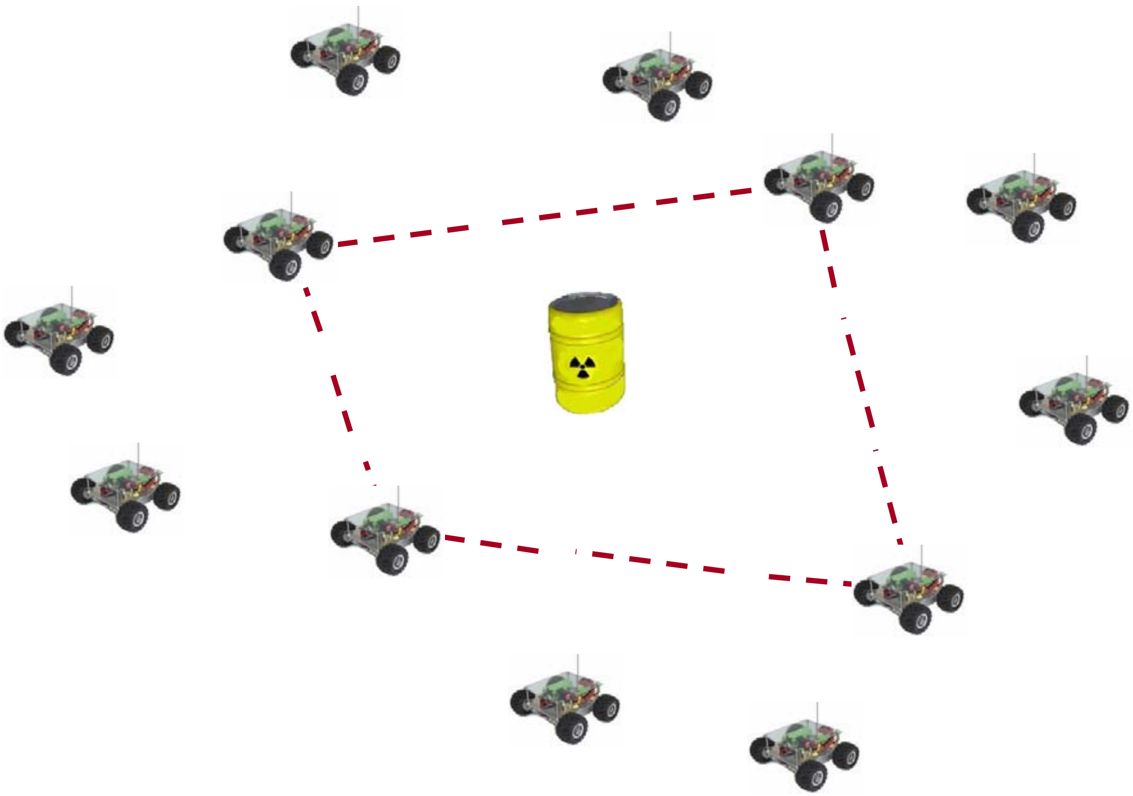
Computer Science Department
California Institute of Technology

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

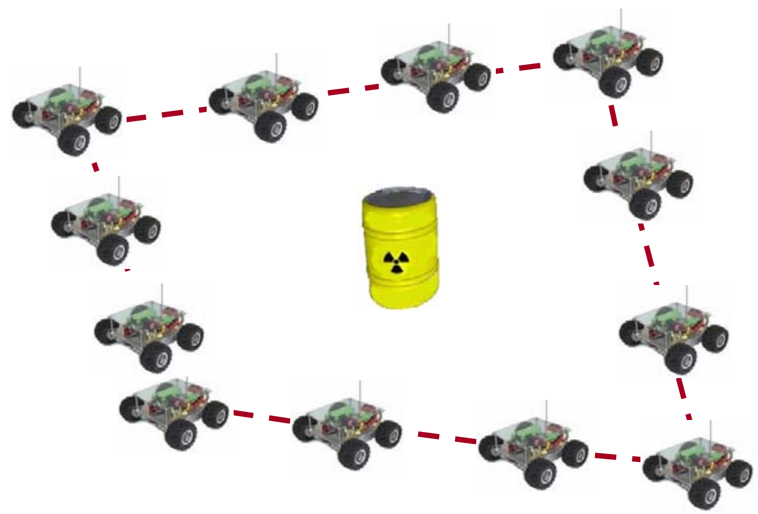
Pattern Formations

- Multi-agent System whose goal is forming "*A fence around a given target*"



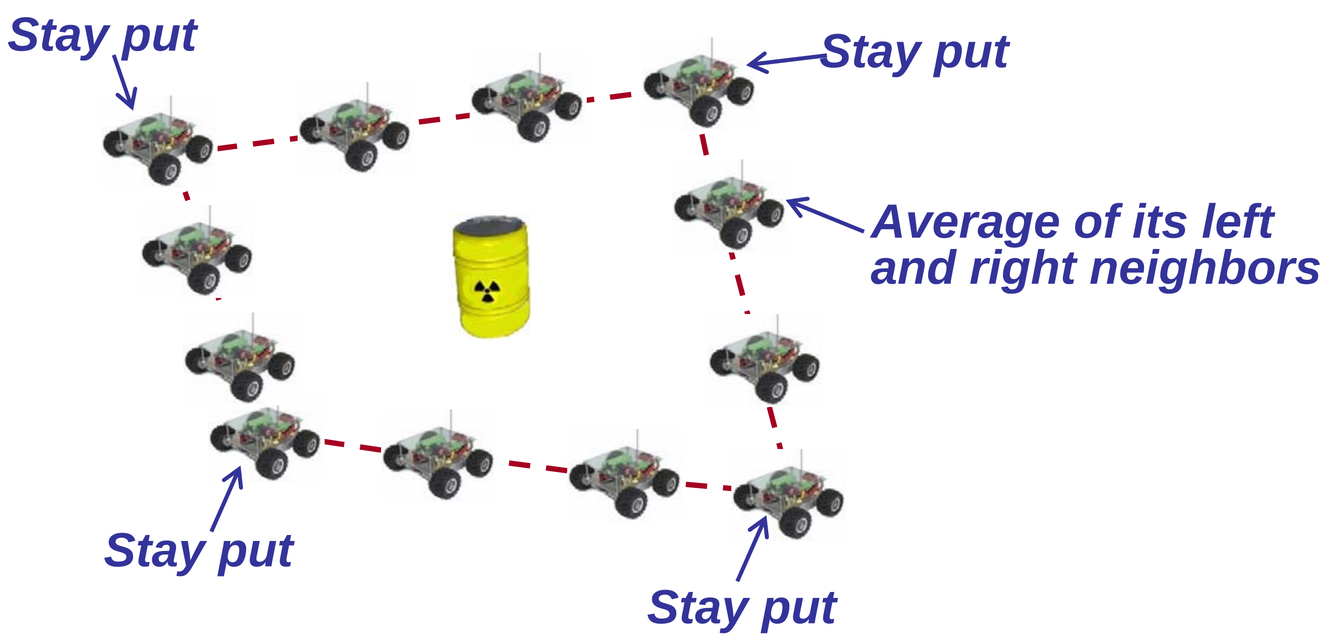
Target Formation

- Multi-agent System whose goal is forming "*A fence around a given target*"
- In the final configuration, robots are equispaced



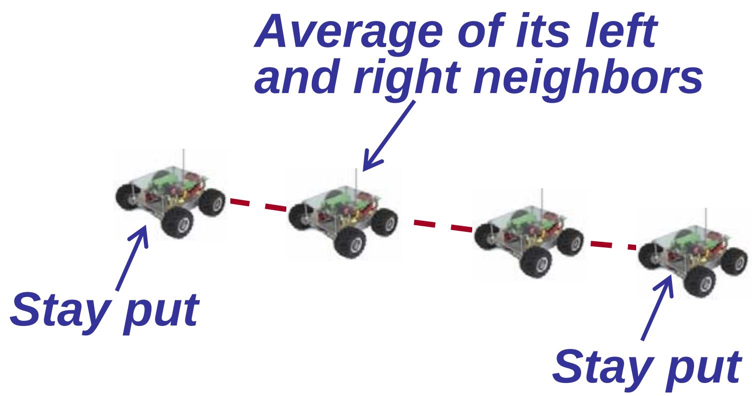
Protocol

- 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



$x_0 = init_0$

$x_1 = 0.5\, x_0 + 0.5\, x_2$

$$
\begin{array}{ccccc}
A & & x & & b \\
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-0.5 & 1 & -0.5 & 0 & 0 & 0 \\
 & & & \cdots & & \\
0 & 0 & 0 & -0.5 & 1 & -0.5 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
& \cdot &
\begin{pmatrix}
x_0 \\ x_1 \\ \cdots \\ x_{N-1} \\ x_N
\end{pmatrix}
& = &
\begin{pmatrix}
init_0 \\ 0 \\ \cdots \\ 0 \\ init_N
\end{pmatrix}
\end{array}
$$

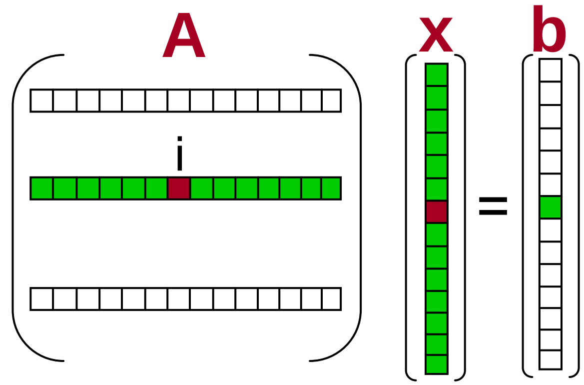
Outline of the Talk

- 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

$$\boxed{A\ x = b}$$

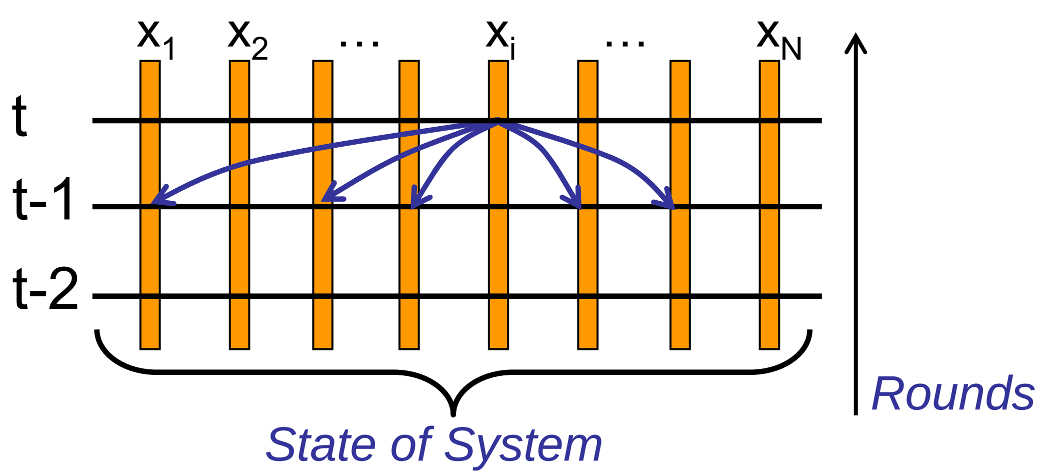


- *Assumptions*:
  - *A* ***invertible*** with ***diagonal*** entries of 1
- *Decentralized System: Multi-Agent System* where agent i is responsible for computing x_i

Gauss Scheme

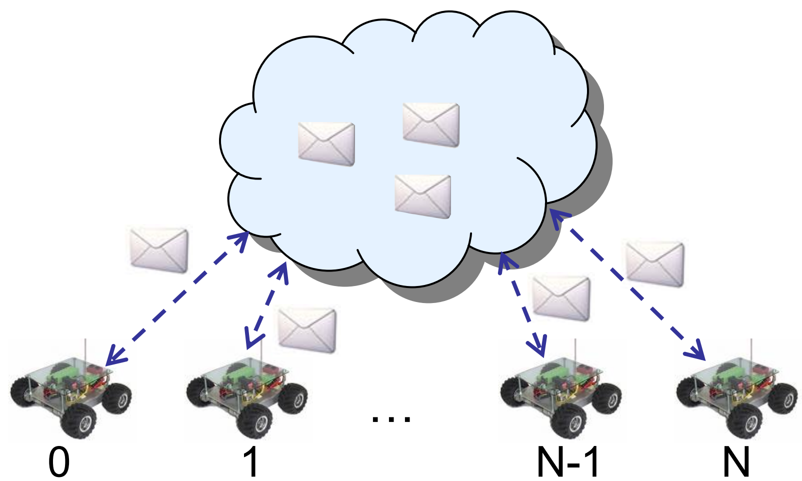
- Solving System of Linear Equations in Rounds starting from an initial guess **init**

$$x_i\,(t) = \begin{cases} b_i - \Sigma_{j\neq i}\, A(i, j)\, x_j\,(t\text{-}1) & t>0 \\ init_i & t=0 \end{cases}$$

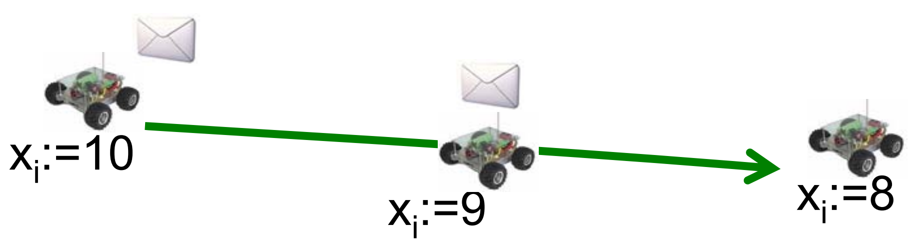


Pattern Formation Challenges

- *Unreliable* Message-Passing Communication

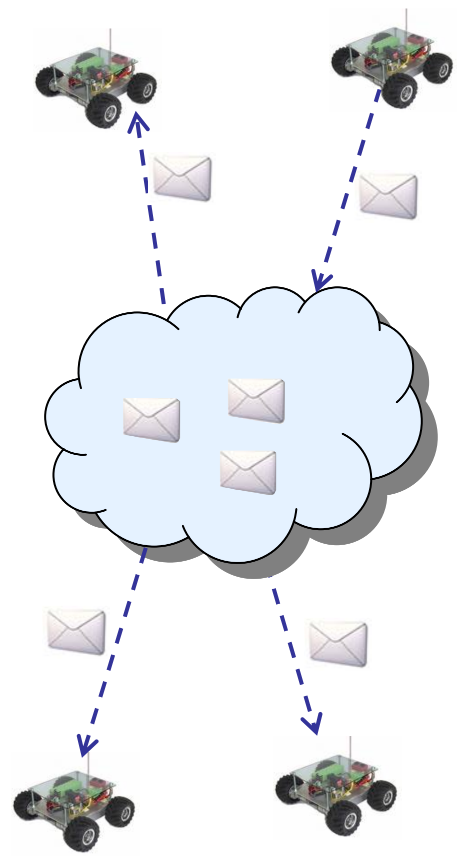


- Agents do not update their positions *instantaneously*



Communication Medium

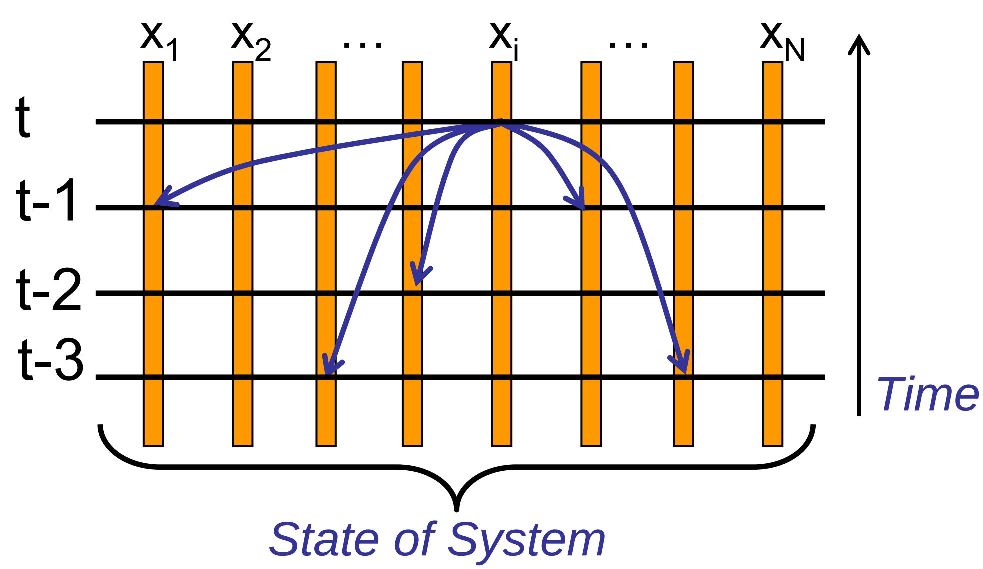
- *Unreliable Broadcast Communication Medium*:
  - Agents send and receive messages
  - Messages in transit can be lost, delayed or received out-of-order
  - Bounded transmission delay
  - Agent i stores in the variable msg_{ij} the last message it receives from agent j
  - Agent i broadcasts x_i infinitely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



Gauss over Unreliable Networks

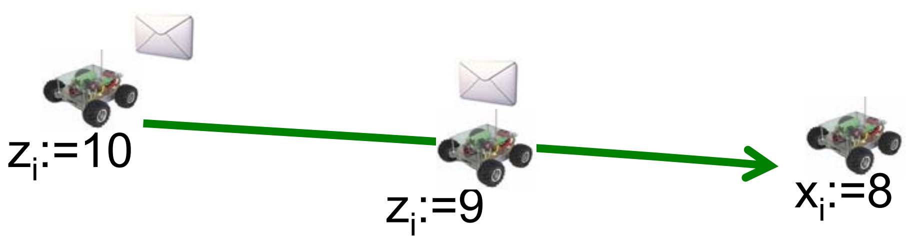
- *Over Message-Passing Systems:*

$$x_i\ (t) = \begin{cases} b_i - \Sigma_{j\neq i} A(i, j)\ msg_{i,j} & t>0 \\ init_i & t=0 \end{cases}$$



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



Does this Scheme work?

If *A* satisfies

(D1) invertible

(D2) diagonal entries equal to 1

$$\boxed{\forall i : A(i,i) = 1}$$

(D3) *weakly diagonally dominant*

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

(D4) *strictly diagonally dominant* in at least one row

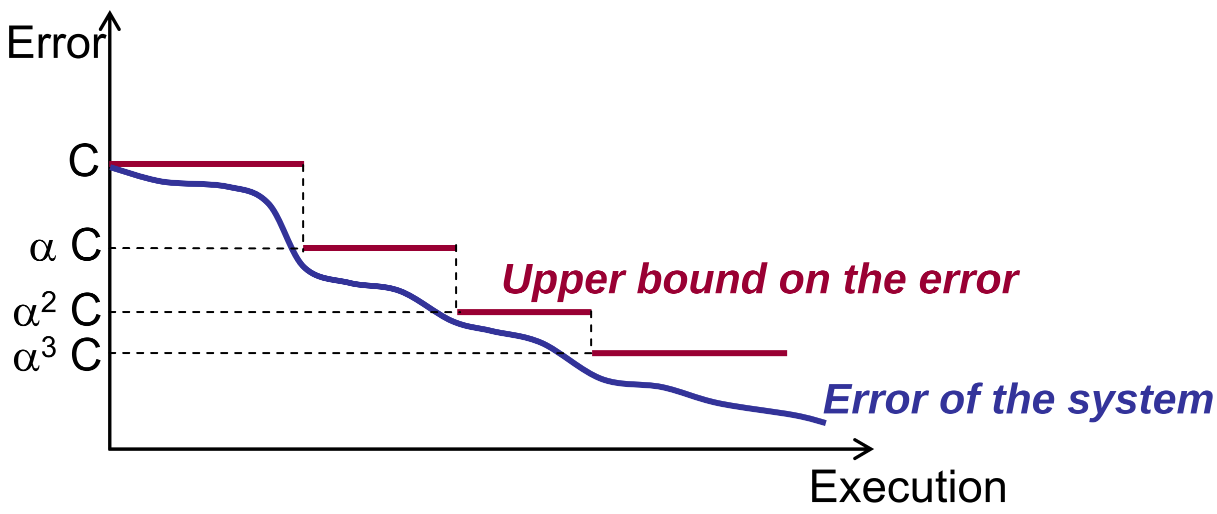
$$\boxed{\exists j \ : \Sigma_{k \neq j} \ | A(j,k) | < 1}$$

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

Proof Summary

(E1) Error of the system E does not increase

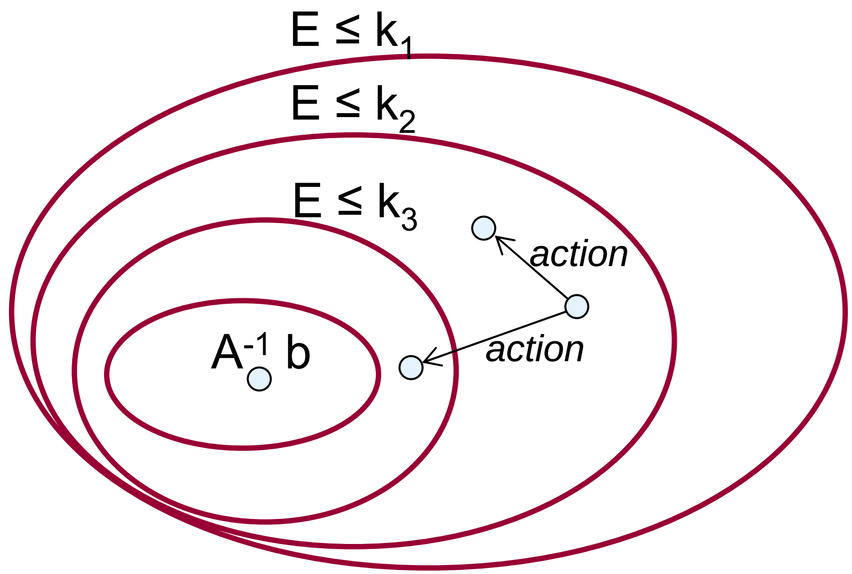
(E2) E eventually decreases by a factor α



Proof Summary

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

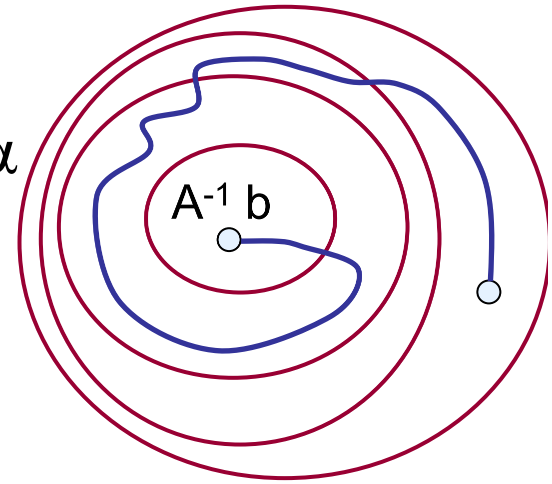
– For all actions of the system, the execution of the action does not increase the error



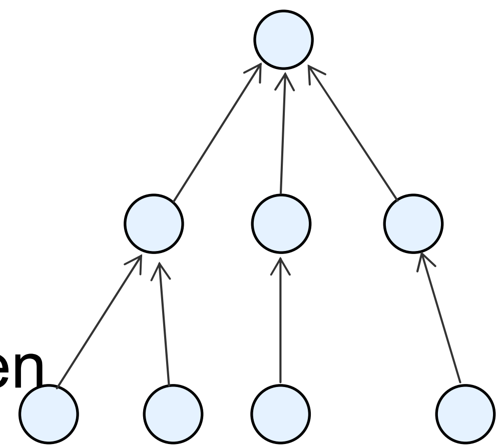
$k_1 > k_2 > k_3 \ldots$

Proof Summary

(E2) E eventually decreases by a factor α



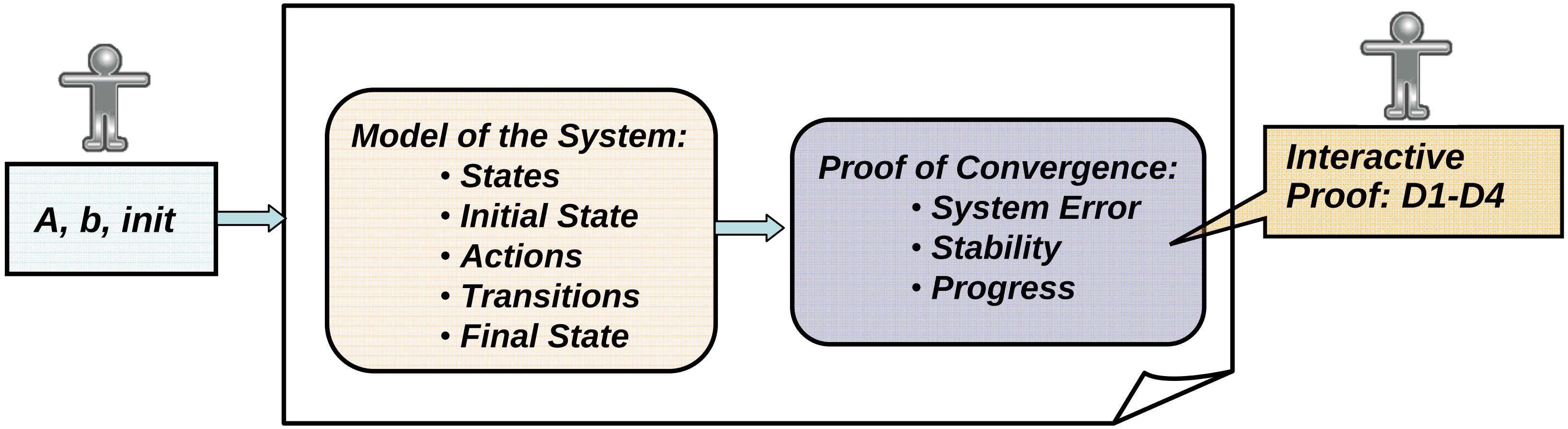
- Proved by *Induction*
- Tree based on communication links in A
  - (i, j) is an edge if $A(i, j) \neq 0$
  - Roots: Strictly Diagonally Dominant agents
- Induction Proof: Assume E = C
  - Base Case: Error of Roots eventually α C
  - Induction Step: assuming error along path to agent j (exclusive of j) is α C then eventually error of j is α C

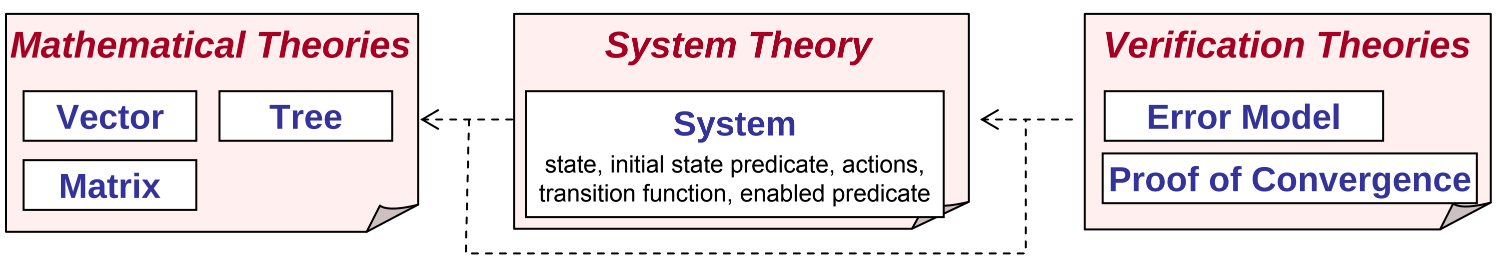


Outline of the Talk

- 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

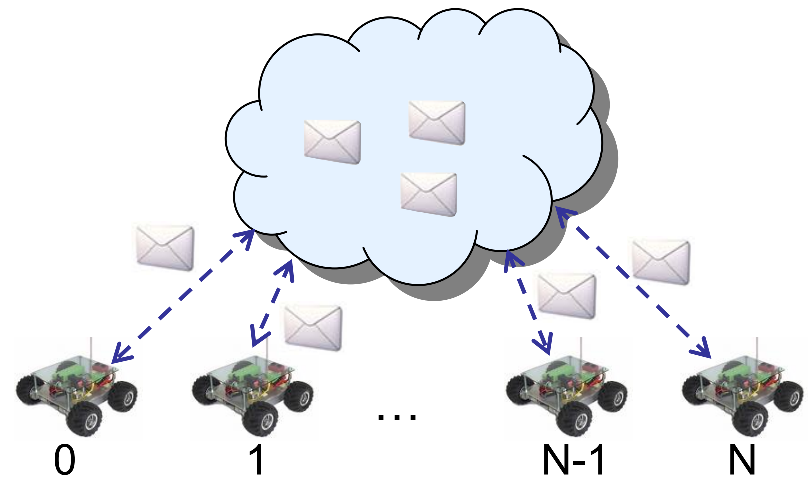
PVS Verification Framework





Description of the System

- Automaton with:
  - States
  - Initial State Predicate
  - Actions
  - Transition Function
  - Enabling Predicate
- Extend time_machine metatheory

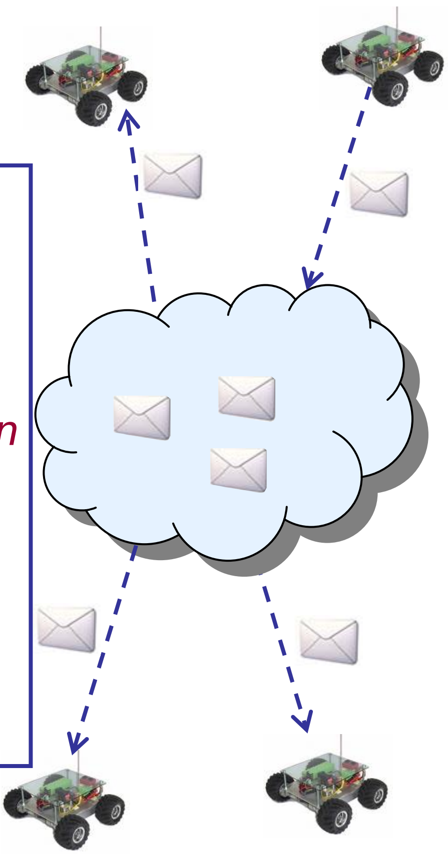



```
IMPORTING time_machine[S,        State
                       ACS,      Actions
                       enabled,  Enabling Predicate
                       trans,    Transition function
                       start?, …] Initial State Predicate
```

State of the System

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

```
S: TYPE = [#

     target : Vector, State of Agent

     lastmsg : Matrix,State of Agent

     buffer : [Index, Index -> Pset], Communication
                                      Medium

     now : nonneg_real, Global Clock

     next: [ Index -> nonneg_real]Communication
#]                                Medium
```



State of the Agents

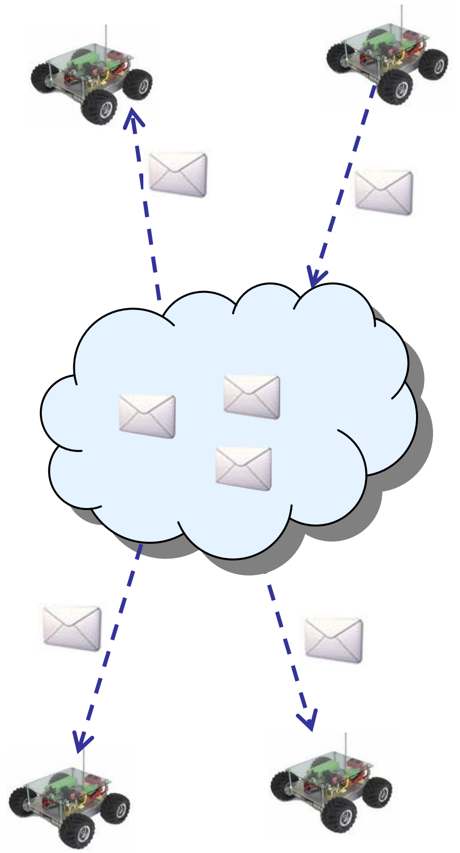
```
Index : TYPE = below(N)
```
Agents' Identifiers

```
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]
#]
```

variable x

variable m and z along diagonal

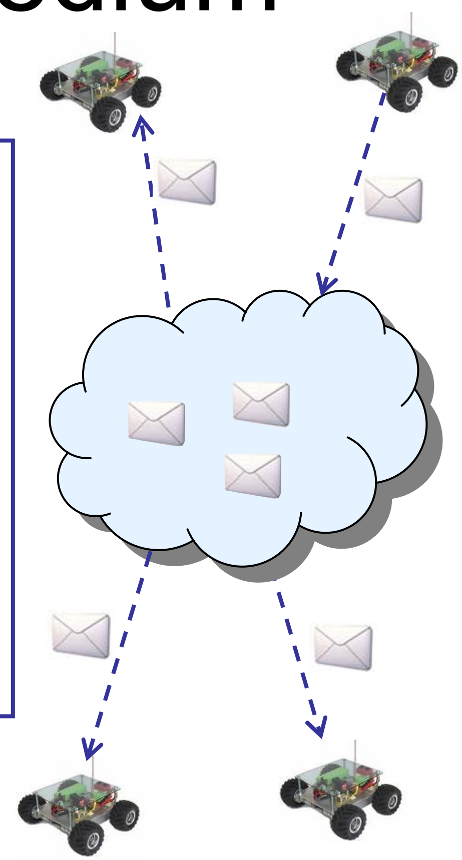


State of Communication Medium

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

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*

```
Msg:  TYPE = [# loc: real, id: Index #]
Pkt:  TYPE = [# msg: Msg, ddl: posreal #]
Pset: TYPE = set[Pkt]
b: posreal
d: posreal
```

Deadline – model bounded delay

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

Maximum delay

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

Initial State Predicate

- Describe properties of the initial state

```
start?(s: S): bool =
    now(s) = 0 Global clock 0
    AND
    target(s)=x0 x initialized with init
    AND
    (FORALL (i:Index): next(s)(i)<= d)
    AND
    (FORALL (i,j: Index): lastmsg(s)(i,j)= x0(j))
```

Schedule the next time an agent execute a send action

$m_{i,j}$ =init$_j$

- 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?Advance Time –Time Manager
    send(p:Pkt, i:Index , d1:posreal): send? Agents
    receive(p:Pkt, i:Index): receive? Agents
    msgloss(p:Pkt, i:Index): msgloss?Packet loss - Channel
    move(i:Index, delta_t:posreal): move?Agents
END ACS
```

Transition Function and Enabling Conditions

- Body of the Actions

```
trans (a:ACS, s:S): S =
CASES a OF
    nu_traj (delta_t): …
    send (p,i,d1): …
    receive (p,i): …
    msgloss (p,i): …
    move (i,delta_t): …
ENDCASES
```

- Enabling Conditions

```
enabled (a:ACS, s:S): bool=
CASES a OF
    nu_traj (delta_t): …
    send (p,i,d1): …
    receive (p,i): …
    msgloss (p,i): …
    move (i,delta_t): …
ENDCASES
```

Time Action

- Action

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

Advance Clock of delta_t unit

- Enabling Conditions

```
enabled (a:ACS, s:S): bool=
CASES a OF
    nu_traj (delta_t):
    FORALL (p: Pkt): ddl(p) >= now(s) + delta_t
ENDCASES
```

Clock can be advanced only if the new time does not violate packet deadlines

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 ]
]
```

Broadcast p along all outgoing channels of i

Schedule the next send

- Enabling Conditions

```
send (p, i, d1): next(s)(i) = now(s)
                 AND d1<=d
                 AND id(msg(p))=i
                 AND  loc(msg(p)) = target(s)(i)
                 AND ddl(p) = now(s) + b
```

Agent is allowed to send

Next scheduled time does not violate d

Send its current value and id

Packet deadline does not violate b

Agent Receive Action

- Agent i receives packet p

```
receive (p:Pkt, i:Index):
LET m: Msg = msg(p),
    j: Index = id(m),
    l: real = loc(m),
    Ci: Vector = update(row(lastmsg(s),i),j,l)
IN s WITH [
    buffer := buffer(s) WITH
           [(j,i) := remove(p,buffer(s)(j,i))],
    lastmsg := lastmsg(s) WITH [ (i,j):= l ],
    target := target(s) WITH [(i):= gauss(Ci,i)]]
```

Sender and location of the message

Remove p from the channel from j to i

update m_{ij}

compute new x_i

- Enabling Conditions

```
receive (p,i):
    buffer(s)(id(msg(p)),i)(p) AND ddl(p) >= now(s)
```

p is in the channel from j to i and its deadline is not violated

Message Loss Action

- Packet p is removed from the System

```
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                                          Remove p from the channel
    WITH [ buffer := buffer (s)
           WITH [ (j,i) := remove(p, buffer(s)(j,i))]]
ENDCASES
```

- Enabling Conditions

```
enabled (a:ACS, s:S): bool=
CASES a OF                             p is in the buffer from sender of p to i
    msgloss (p,i): buffer(s)(id(msg(p)),i)(p)
ENDCASES
```

Agent Move Action

- Agent i moves from its current to its target value

```
move (i:Index , delta_t:posreal):
s WITH [                                        Executes a move action
    lastmsg := lastmsg(s) WITH [(i,i) := target(s)(i)],
    now := now(s) + delta_t  Advance Clock of delta_t unit
]
```

- Enabling Conditions

```
move (i:Index , delta_t:posreal): No packet deadline is violated
    FORALL (p: Pkt): ddl (p) >= now (s) + delta_t
```

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

```
dd?(m): bool =
FORALL (r: Index): sum(row(abs(m),r),r) <= abs(m(r,r))
sdd?(m): bool =
EXISTS (r: Index): sum(row(abs(m),r),r) < abs(m(r,r))
```

Error Model

- Error of the system as the maximum of agent errors

```
me : [S -> nonneg_real]
me_all_error: AXIOM FORALL(i) : mes(s,i)<= me(s)
me_ex_error:  AXIOM EXISTS(i) : mes(s,i) = me(s)
```

- Error of agent i in the system

```
mes(s,i): nonneg_real = max(mae(s,i), mbe(s,i))
```

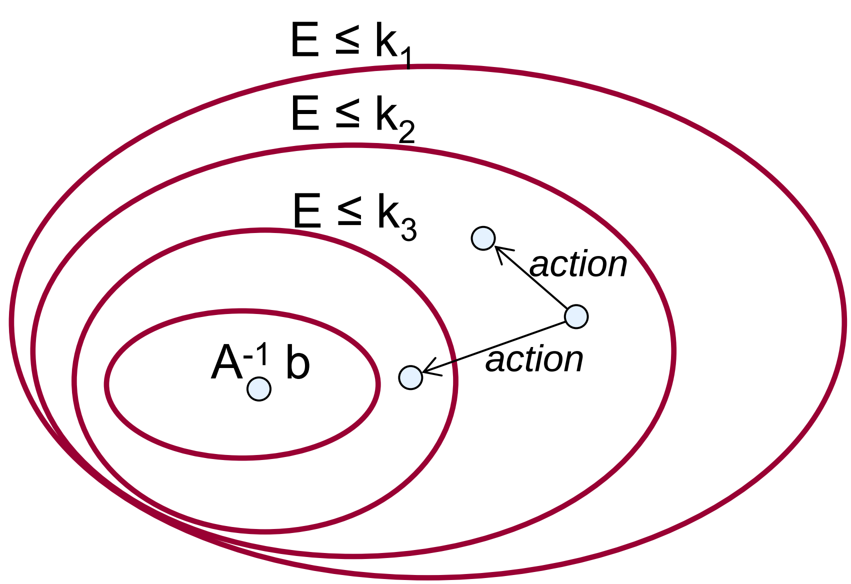
Error of i is the maximum of its error along its outgoing channels and its error in all agents

Proof Summary

(E1) Error of the system E does not increase

```
not_incr_error :
LEMMA enabled(a,s) IMPLIES me(s) >= me(trans(a,s))
```

- Proof is a case on the actions



$k_1 > k_2 > k_3 \ldots$

Proof Summary

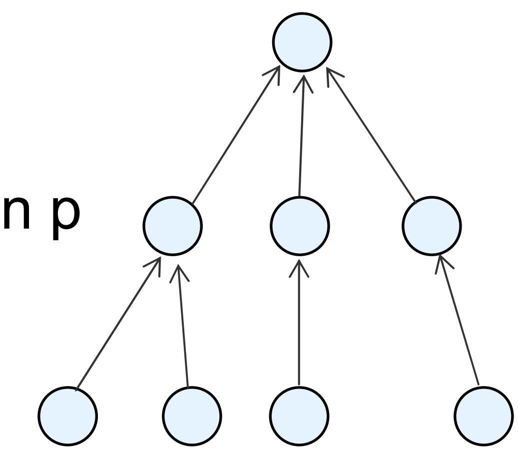
(E2) E eventually decreases by a factor α

– p defined recursively along the tree t

```
p(i:Index) RECURSIVE nonneg_real =
  IF root_t?(t,i)
  THEN sum(row(abs(A),i),N-1,i)          Sum along the row
  ELSE abs(A(i,parent(t,i))) * p(parent(t,i)) +
         sum(row(abs(A),i),N-1,i,parent(t,i))
  ENDIF                       Sum along the row and path to the root
```

– showed that $0 \le p < 1$

– α defined as the maximum of function p



Proof Summary

(E2) E eventually decreases by α

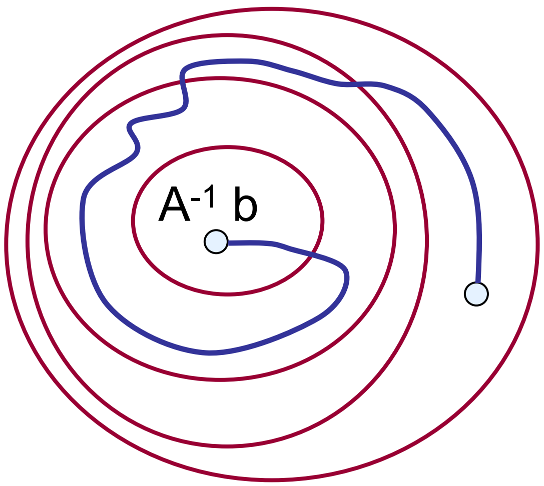
- Define a family of predicates Z
- Prove Stability

```
Z_stable:
LEMMA Z(s,C,i) AND enabled(a,s)
IMPLIES Z(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

A, b, init

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

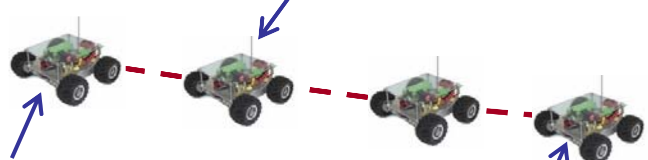
Proof of Convergence:
- ***System Error***
- ***Stability***
- ***Progress***

Interactive Proof: D1-D4

$$A=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 1 & -0.5 & 0 & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & -0.5 & 1 & -0.5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$b=\begin{pmatrix} init_0 \\ 0 \\ \dots \\ 0 \\ init_N \end{pmatrix}$$

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