Convergence Verification: From Shared Memory to Partially Synchronous Systems*

K. Mani Chandy, Sayan Mitra, and Concetta Pilotto

California Institute of Technology
Pasadena, CA 91125
{mani,mitras,cetta}@caltech.edu

Abstract. Verification of partially synchronous distributed systems is difficult because of inherent concurrency and the potentially large state space of the channels. This paper identifies a subclass of such systems for which convergence properties can be verified based on the proof of convergence for the corresponding discrete-time shared state system. The proof technique extends to the class of systems in which an agent's state evolves continuously over time. The proof technique has been formalized in the PVS interface for timed I/O automata and applied to verify convergence of a mobile agent pattern formation algorithm.

1 Introduction

In a partially synchronous distributed system a collection of processes interact by exchanging messages. Sent messages are either lost or delivered within a constant but unknown time bound. This model of communication presents an interesting and realistic middle-ground between the two extremes of completely synchronous (lock-step execution) and asynchronous (unbounded message delay) models. The model is particularly appropriate for a wide class of systems including those employing wireless communication and mobile agents. Algorithms and impossibility results for problems such as mutual exclusion and consensus [10] in this model have been studied extensively (see, for example, Chapters 24-25 of [16] and the bibliographic notes).

Partially synchronous systems are difficult to understand and reason about because of their inherent concurrency and message delays. Formal models, in particular variants of Timed Automata [2,13], have been used to model and analyze such systems, however, there have been few applications of formal verification techniques in checking correctness. Typically these systems present difficulty for model checking because of the huge state space which includes the (potentially large number of) messages in transit. Nevertheless, in a recent paper [11] the time to reach agreement of a consensus protocol has been model checked with UPPAAL [5] by exploiting a key compositional property of the protocol. Two other partially synchronous distributed algorithms have been model checked in [14].

* The work is funded in part by the Caltech Information Science and Technology Center and AFOSR MURI FA9550-06-1-0303.

In this paper, we study partially synchronous distributed systems (with possibly continuous state spaces), with the aim of verifying *convergence*. Such systems arise in sensor networks, mobile robotics, and unmanned vehicle coordination applications, and convergence properties capture the requirement that the distributed system iteratively computes a certain function. For example, the requirement that a set of mobile robots get arbitrarily close to a particular spatial pattern through communication is a convergence property.

Techniques based on analyzing the Eigen values of state-transition matrices [18,6] that have been used for verifying convergence of completely synchronous systems, cannot be applied in a straightforward way to highly nondeterministic partially synchronous systems. The main contributions of this paper are: (i) a methodology for transforming a shared state distributed system—in which processes can read each other's state instantaneously—to a corresponding partially synchronous system, such that the convergence properties of the original system are preserved in the latter, (ii) a substantial verification case study carried out within the Tempo/PVS framework [4,1] based on the above theory.

We begin in Section 2 by describing *Shared State (SS)* systems—a general discrete-time model for distributed systems in which each process can change its state by reading the states of some subset of other processes. A change of state can be nondeterministic and each process is free to change its state at any point in time, independent of the others. We adapt a theorem from Tsitsiklis [20], to obtain a sufficient condition for proving convergence of such shared state systems. Given a shared state system $\mathcal{A}$, this sufficient condition requires us to find a collection of shrinking invariant sets for $\mathcal{A}$. Next, in Section 3, we present a natural transformation of the given shared state system $\mathcal{A}$ to a partially synchronous system $\mathcal{B}$. The partially synchronous system is modeled as a Timed Input/Output Automaton [13]. In Section 4, we show that if $\mathcal{A}$ converges, then under some assumptions about the structure of the invariant sets of $\mathcal{A}$ and message losses in $\mathcal{B}$, $\mathcal{B}$ also converges. Our proof relies critically on properties of the collection of shrinking invariants that are used in the theorem of [20].

In Section 5, we apply the above theory to verify convergence of a partially synchronous pattern formation protocol for mobile agents. First, we specify the shared state version of the protocol in PVS and verify its convergence using the pre-existing PVS metatheory [17] . We obtain the partially synchronous version of the pattern formation system; this is specified in PVS using the PVS/TIOA toolset [15] and we show that it satisfies the assumptions required for convergence.

2 Preliminaries

In this section we present a standard discrete-time model for shared state distributed systems and state a well-known theorem for proving convergence.

Standard notations are used for natural numbers $\mathbb{N} = \{0, 1, \ldots, \}$ and the set of reals $\mathbb{R}$. For $N \in \mathbb{N}$, the set $\{0, 1, 2, \ldots, N\}$ is denoted by $[N]$. For a set A, $A_\perp \triangleq A \cup \{\perp\}$. The set of finite sequences of length N (and infinite sequences)

of elements in A is denoted by A^N (and resp., A^ω). For $a \in A^N, i \in [N-1]$, the i^{th} element of a is denoted by a_i. The same notation is used for infinite sequences. For any $x \in A, i \in [N], a \in A^{N+1}$, $[a|a_i := x]$ denotes the (unique) element $a' \in A^{N+1}$ satisfying: for all $j \in [N]$, if $j = i$ then $a'_j = x$ else $a'_j = a_j$.

A *Labeled Transition System* $\mathcal{A}$ is a quadruple $(S, S_0, A, \rightarrow)$ where (a) S is a set of *states*, (b) $S_0 \subseteq \mathcal{S}$ is a set of *start states*, (c) A is a set of *actions*, and (d) $\rightarrow \subseteq S \times A \times S$ is a set of *transitions*. For $(s, a, s') \in \rightarrow$ we write $s \xrightarrow{a} s'$. An *execution* α of $\mathcal{A}$ is an (finite or infinite) alternating sequence of states and actions $s_0 a_1 s_1 a_2 \ldots$, such that $s_0 \in S_0$ and for all i, $s_i \xrightarrow{a_{i+1}} s_{i+1}$. An LTS is said to be *action deterministic* if for any $s, s', s'' \in S, a \in A$, if $s \xrightarrow{a} s'$ and $s \xrightarrow{a} s''$ then $s' = s''$. Thus, each action $a \in A$ is associated with a unique *state transition function* $f_a : S \rightarrow S$, such that if $s \xrightarrow{a} s'$ then $s' = f_a(s)$.

Convergence. In order to define convergence of an execution to a state $s^* \in S$ we have to introduce some notion of "closeness" of states to s^*. One straightforward way to do this, and the approach we take in presenting this paper, is to assume that S is equipped with a metric d. An infinite execution α *converges* to s^* with respect to d, if for every $\epsilon > 0$, there is a suffix of α such that for every state s in this suffix $d(s, s^*) \leq \epsilon$. Convergence to a subset $S^* \subseteq S$ is defined by extending the definition of d in the obvious way. We remark that for defining convergence to s^* or to a subset S^* of S, it is not necessary for S to be a metric space, and it suffices to have a topological structure around s^* (or S^*). The results presented in this paper carry over to this more general setting.

For verifying convergence, we restrict our attention to executions in which certain classes of actions occur infinitely often. This motivates the notion of fair executions. For a set of actions A, a *fairness condition* $\mathcal{F}$ is a finite collection $\{F_i\}_{i=1}^n$, $n \in \mathbb{N}$, where each F_i is a nonempty subset of A. An infinite sequence of actions $a \in A^\omega$ to be *$\mathcal{F}$-fair* iff $\forall\ F \in \mathcal{F}, n \in \mathbb{N},\ \exists\ m \in \mathbb{N},\ m > n,$ such that $a_m \in F$. An infinite execution $\alpha = s_0, a_0, s_1, a_1, \ldots$ is *$\mathcal{F}$-fair* exactly when the corresponding sequence of actions $a_0, a_1, \ldots$ is $\mathcal{F}$-fair. Under a given fairness condition $\mathcal{F}$, an LTS $\mathcal{A}$ is said to *converge* to s^* if every $\mathcal{F}$-fair execution converges to s^*.

Usually a convergence proof is carried out by showing the existence of a Lyapunov-like function that is nonnegative and decreases along all executions of the system. The following theorem from [20], translated to our setting, provides a general sufficient condition for proving convergence in terms of a collection of invariant sets (sublevel sets of a Lyapunov function).

Theorem 1. *Consider an LTS $\mathcal{A}$ and a fairness condition $\mathcal{F}$ for $\mathcal{A}$. Suppose there exists a well ordered set $(T, <)$ with smallest element 0 and a collection of sets $\{P_k \subseteq S |\ k \in T\}$ satisfying:*

C1. (Monotonicity) $\forall\ k, l \in T, k > l \Rightarrow P_k \subsetneq P_l$.
C2. (Granularity) $\forall\ \epsilon > 0, \exists\ k \in T$, such that $\forall s \in P_k, d(s, s^*) \leq \epsilon$.
C3. (Initial) $S_0 \subseteq P_0$.
C4. (Invariance) $\forall\ s, s' \in S, a \in A, k \in T$ if $s \xrightarrow{a} s'$ and $s \in P_k$ then $s' \in P_k$.

C5. (Progress) $\forall\ k \in T$, *if* $P_k \neq \{s^*\}$ *then* $\exists\ F \in \mathcal{F}$, *such that* $\forall\ a \in F$, $\forall\ s \in P_k, s' \in S$, $s \xrightarrow{a} s' \ \Rightarrow\ s' \in P_l$, *for some* $l > k$.

Then all $\mathcal{F}$*-fair executions of* $\mathcal{A}$ *converge to* s^* *with respect to* d.

It turns out that under some weak assumptions about the stability of $\mathcal{A}$, these conditions are also necessary for convergence of $\mathcal{A}$. C1 requires that the sequence of predicates is monotonically stronger. C2 states that for every $\epsilon > 0$ there exists a a set P_k that is contained in the ϵ-ball around s^*. C4 requires that the P_k's are invariant under the transitions of $\mathcal{A}$. Finally, C5 requires that for any state s in P_k (other than s^*) there exists a fair set F in $\mathcal{F}$, such that any action in F takes the system to a state P_l, where $l > k$.

Shared State Systems. A distributed system consists of a finite collection of LTSs executing and communicating in parallel. In a shared state (distributed) system a process can read but not modify the states of other asynchronous processes. Formally, a *shared state distributed system* with $N+1$ processes is an action deterministic LTS $(S, S_0, A, \rightarrow)$ with the following additional structure:

(a) $S \triangleq X^{N+1}$, where X is a set of *process states*. For each $s \in S, i \in [N]$, s_i is called the state of the i^{th} process.
(b) $S_0 = \{x_0\}$, where $x_0 \in X^{N+1}$ is the vector of initial SS process states,
(c) The set of actions A is partitioned into disjoint sets $\{A_i\}_{i\in[N]}$ such that for all $s, s' \in S, a \in A_i$, if $s \xrightarrow{a} s'$ then $\forall\ j \in [N] \setminus \{i\}$, $s_j = s'_j$.

An action $a \in A_i$ corresponds to process i reading the current states of a subset of other agents and updating its own state. For each action $a \in A_i$ we denote the state transition function f_a restricted to the i^{th} component (mapping X^{N+1} to X) by f_{ia}. That is, if $s \xrightarrow{a} s'$ then $s' = [s|s_i = f_{ia}(s)]$. Function f_{ia} is a function of the states of some subset of processes and is independent of the states of other processes; this is captured by the *dependency function* $D : A \rightarrow 2^{[N]}$ as follows: for any pair of states $s, u \in S$, $i \in [N]$, and any action $a \in A$, if for all $j \in D(a)$, $s_j = u_j$ then the $f_{ia}(s) = f_{ia}(u)$. That is, the post-state of action a depends on the j^{th} state component of the pre-state only if $j \in D(a)$. We say that j is a *neighbor* of i exactly when there exists $a \in A_i$ such that j is in $D(a)$.

3 Partially Synchronous Systems

In this section, we present the model for partially synchronous distributed systems and describe a natural translation of shared state systems to this model. In a partially synchronous distributed system a fixed set of processes communicate by sending messages over a broadcast channel. A message broadcast by process i at some time t is delivered to some (possibly empty) subset of processes; all (if any) deliveries are within $t + b$, where b is a parameter of the broadcast channel.

Timed I/O Automata. We formally model partially synchronous distributed systems as Timed Input/Output Automata (TIOA) [13]. A Timed I/O Automaton is a non-deterministic state transition system in which the states may change either (a) instantaneously through a transition, or (b) continuously over an interval of time following a *trajectory*. We give the essential definitions for the TIOA framework and refer the reader to [13] for the details. A variable structure is used to specify the states of a TIOA. Let V be a set of variables. Each variable $v \in V$ is associated with a type which defines the set of values v can take. The set of valuations of V is denoted by $val(V)$. A *trajectory* for a set of variables V models continuous evolution of values of the variables. Formally, a trajectory τ maps a left-closed interval of $\mathbb{R}_{\geq 0}$ with left endpoint 0 to $val(V)$. The domain τ is denoted by $\tau.dom$. A trajectory is *closed* if $\tau.dom = [0, t]$ for some $t \in \mathbb{R}_{\geq 0}$, in which case we define $\tau.\mathsf{ltime} \triangleq t$ and $\tau.\mathsf{lstate} \triangleq \tau(t)$.

A TIOA $\mathcal{B} = (V, S, S_0, A, \mathcal{D}, \mathcal{T})$ consists of (a) A set V of *variables*. (b) A set $S \subseteq val(V)$ of *states*. (c) A set $S_0 \subseteq S$ of *start states*. (d) A set A of *actions* partitioned into *input, output* and *internal* actions I, O, and H, (e) A set $\mathcal{D} \subseteq S \times A \times S$ of *discrete transitions*. An action $a \in A$ is said to be *enabled* at s iff $(s, a, s') \in \mathcal{D}$. (f) A set $\mathcal{T}$ of trajectories for V that is closed[1] under prefix, suffix and concatenation. In addition, for every $\mathbf{s} \in S$, $\mathcal{A}$ must satisfy the following two nonblocking conditions: (i) $\forall a \in I$, a is enabled at $\mathbf{s}$, and (ii) $\exists \tau \in \mathcal{T}$, such that $\tau(0) = \mathbf{s}$ and either $\tau.dom = [0, \infty)$ or τ is closed and $\exists a \in O \cup H$ enabled at $\tau.\mathsf{ltime}$.

An *execution fragment* of $\mathcal{B}$ is a finite or infinite alternating sequence of trajectories and actions $\tau_0 a_1 \tau_1 a_2 \ldots$, such that for all i in the sequence, $\tau_i.\mathsf{lstate} \overset{a_{i+1}}{\rightarrow} \tau_{i+1}(0)$. We define the first state of α, to be $\alpha.\mathsf{fstate} \triangleq \tau_0(0)$, and for a closed α, its last state to be $\alpha.\mathsf{lstate} \triangleq \tau_n.\mathsf{lstate}$, where τ_n is the last trajectory in α, and $\alpha.\mathsf{ltime} \triangleq \sum_i \tau_i.\mathsf{ltime}$. An execution fragment α is *admissible* if $\alpha.\mathsf{ltime} = \infty$. An execution fragment is an *execution* if $\tau_0(0) \in S_0$.

Given a shared state system $\mathcal{A} = (S, x_0, A, \rightarrow)$ for processes indexed by $[N]$ we define a natural translation of $\mathcal{A}$ to the partially synchronous setting. The partially synchronous system corresponding to a given shared state system $\mathcal{A}$ is a TIOA $\mathcal{B}$ obtained by composing a set of $\mathsf{Process}_i$ TIOAs—one for each $i \in [N]$—and an TIOA LBCast which models the communication channels.

Generic process. First, we specify a TIOA $\mathsf{Process}_i$ for each participating process $i \in [N]$. The code in Figure 1 specifies this automaton using the TIOA Language [12]. The specification is parameterized by (a) an uninterpreted type X, (b) a element x_{0i} of X representing the initial state of process i, (c) a collection of functions $g_{ia} : X_{\perp}^{N+1} \rightarrow X$, for $i \in [N], a \in A_i$ representing the actions of i, and (d) nonnegative real-valued parameters l and w dealing with timing. In order to obtain the process corresponding to $\mathcal{A}$, these parameters are instantiated as follows: (i) the type X equals the process state set of $\mathcal{A}$, (ii) x_{0i} is set

[1] See Sections 3-4 of [13] for formal definitions of the trajectory closure properties and the statements of the enabling conditions.

to the i^{th} component of the start state of $\mathcal{A}$, (iii) for each $a \in A_i$, and for any $x : X, y : \texttt{Array}[[N] \to X_\perp]$

$$g_{ia}(x, y) \triangleq \begin{cases} f_{ia}([y|y_i := x]) \text{ if } \forall j \in D(f_{ia}), y[j] \neq \perp \\ x \qquad\qquad\qquad\quad \text{otherwise.} \end{cases}$$

$\mathsf{Process}_i$ has the following state **variables**: (a) x_i is a discrete variable of type X and is initialized to x_{0i} (b) y_i records state information about other processes received through messages. It is an array of type $X_\perp$ indexed by $[N]$ and initialized to $\perp$; $y_i[j]$ is the last message (if any) that i received from j (c) now_i, a continuous variable of type $\mathbb{R}_{\geq 0}$ and initialized to 0, models real time, and (d) $earliest_i$, a discrete variable of type $\mathbb{R}_{\geq 0}$ and initialized to l, is the earliest time for the next broadcast by process i. The initial state is defined by the initial valuations of the variables.

The **transitions** for send_i and $\mathsf{receive}_{ij}$ actions are specified in **pre**condition-**eff**ect style in lines 15–23. (a) $\mathsf{receive}_{ij}(m)$ models the delivery of message m to $\mathsf{Process}_i$ from $\mathsf{Process}_j$ over the broadcast channel. When this action occurs, the j^{th} component of the history variable y_i is updated to m, and the state variable x_i is updated according to a nondeterministically chosen function g_{ia}. (b) A $\mathsf{send}_i(m)$ action models the broadcasting of message m. This action *can* occur whenever $x_i = m$ and now exceeds $earliest_i$. When this action does occurs, $earliest_i$ is advanced to $now_i + l$.

Finally, the state of $\mathsf{Process}_i$ changes over an interval of time according to the **trajectories** specified in lines 11–13. Along any trajectory, x_i and $earliest_i$ remain constant and now_i increases monotonically at the same rate as real-time. The **stop when** condition states that no trajectory continues beyond the time point at which now_i equals $earliest+w$. This forces the trajectory to stop, which along with condition (ii) in the definition of TIOA forces a send to occur.

```
signature                                                    1
  output send_i(m : X)                                       2
  input receive_ij(m : X), where j ∈ [N]                     3
                                                             4
variables                                                    5
  x_i : X := x_0i;                                           6
  y_i : Array[[N] → X_⊥]                                     7
    initially ∀j ∈ [N], y[j] := ⊥                            8
  earliest_i : ℝ_≥0 := l; now_i : ℝ_≥0 := 0                  9
                                                             10
trajectories                                                 11
  evolve d(now_i) = 1                                        12
  stop when now_i ≥ earliest_i + w                           13
                                                             14
transitions                                                  15
  input receive_ij(m)                                        16
  eff y_i[j] := m;                                           17
    let a := choose A_i                                      18
    x_i := g_ia(x, y)                                        19
                                                             20
  output send_i(m)                                           21
  pre m = x_i ∧ now_i ≥ earliest_i                           22
  eff earliest_i := now_i + l                                23
```

Fig. 1. $\mathsf{Process}_i$ TIOA with parameters $X, x_0, A_i, \{g_{ia}\}_{a \in A_i}, l, w : \mathbb{R}_{\geq 0}$.

Channel. The LBCast automaton of Figure 2 specifies the local broadcast-based communication layer of the system. For any $b \in \mathbb{R}_{\geq 0}$, $\mathsf{LBCast}(b)$ ensures that any

message sent by $\mathsf{Process}_i$ at time t is received by some subset of other processes within $[t, t+b]$.

A *timed message* is a pair consisting of a message of type X and a deadline of type $\mathbb{R}_{\geq 0}$. For a timed message p, the message and the deadline are denoted by $p.msg$ and $p.dl$. LBCast has two state variables: (a) *buffer* is a two dimensional array of sets of timed messages; it is initialized to be empty. $buffer[i,j]$ is the set of messages (time stamped with a deadline) sent by i to j, that are in transit. (b) *now* is a continuous variable of type $\mathbb{R}_{\geq 0}$ and it models real time.

The state of LBCast changes through the occurrence of send, receive, and drop actions as follows: (a) $\mathsf{receive}_{ij}(m)$ models the delivery of message m sent by $\mathsf{Process}_j$ to $\mathsf{Process}_i$. This action can occur when there exists $dl \in \mathbb{R}_{\geq 0}$ (actually $\geq now$) such that the timed message $\langle m, dl\rangle$ is in $buffer[j,i]$. As a result of this action *some* message m (with deadline $dl' \geq now$) is removed from $buffer[j,i]$. (b) $\mathsf{send}_i(m)$ models the broadcasting of message m by $\mathsf{Process}_i$. The effect of this action is that the timed message $\langle m, now + b\rangle$ is added to $buffer[i,j]$ for every $j \in \mathcal{I}$. (c) $\mathsf{drop}_{ij}(m)$ models the loss of message m in transit from i to j. This action is enabled as long as the message m is in transit, and the effect is that the message is removed from $buffer[i,j]$.

Along any trajectory of LBCast (see lines 26–29), *buffer* remains constant and *now* increases monotonically at the same rate as real-time. The **stop when** condition enforces the delivery deadline of non-dropped messages by forcing the receive actions to occur.

```
1  signature
2    input send_i(m : X), where i ∈ [N]
3    output receive_ij(m : X), where i, j ∈ [N]
4    internal drop_ij(m : X, dl : ℝ≥0)
5
6  variables
7    buffer : Array[i, j : [N], Set[X × ℝ≥0]] := {}
8    now : ℝ≥0 := 0
9
10 transitions
11   output receive_ij(m)
12   pre ∃ dl : ℝ≥0, ⟨m, dl⟩ ∈ buffer[j, i]
13   eff dl' := choose
14         {dl ∈ ℝ≥0 | ⟨m, dl⟩ ∈ buffer[j, i]}
15       buffer[j, i] := buffer[j, i] \ ⟨m, dl'⟩;
16
17   input send_i(m)
18   eff for j ∈ [N] do
19     buffer[i, j] := buffer[i, j] ∪ ⟨m, now + b⟩
20   od
21
22   internal drop_ij(m, dl)
23   pre ⟨m, dl⟩ ∈ buffer[i, j] ∧ dl ≥ now
24   eff buffer[i, j] := buffer[i, j] \ ⟨m, dl⟩
25
26 trajectories
27   evolve d(now) = 1
28   stop when ∃ m : X, dl ∈ ℝ≥0, i, j : I,
29     ⟨m, dl⟩ ∈ buffer[i, j] ∧ dl = now
```

Fig. 2. $\mathsf{LBCast}_{i,j}$ TIOA with parameter $X, b : \mathbb{R}_{\geq 0}$.

Complete system. The partially synchronous system corresponding to $\mathcal{A}$ is the composed TIOA $\mathcal{B} = \|_{i\in[N]}\mathsf{Process}_i \| \mathsf{LBCast}$. Let the set of states of $\mathcal{B}$ be $\mathcal{S}$. The values of the real-time related variables such as now_i's $earliest_i$, diverge along the admissible executions of $\mathcal{B}$. In studying convergence of $\mathcal{B}$ we are really interested in the behavior of the x_i and the y_i variables and the messages in *buffer* without their time stamps. Hence, we define a projection function *untime*: for any state $s \in \mathcal{S}$, $untime(s)$ is an object that is identical to s except that the components corresponding to $now, now_i, earliest_i$ are removed, every timed

message p is replaced by $p.msg$, and all $\perp$ values are removed from the history variables y_i's. We denote this projected state space of $\mathcal{B}$ by $S_{\mathcal{B}}$ and its elements by $\mathbf{s}, \mathbf{u}$. Each $\mathbf{s} \in S_{\mathcal{B}}$ corresponds to a particular valuation for each non-time-related state variable of B. These variable valuations are denoted by the usual (.) notation. For example, the valuations of the variables x_i and *buffer* at a state $\mathbf{s}$ are denoted by $\mathbf{s}.x_i$ and $\mathbf{s}.buffer$. We define a metric on $S_{\mathcal{B}}$ based on the metric d on $S_{\mathcal{A}}$ as follows:

$$U(\mathbf{s}) \triangleq \Pi_{i=0}^{N} \left\{ \{\mathbf{s}.x_i\} \cup_{j \in [N]} \{\mathbf{s}.y_j[i] \mid \mathbf{s}.y_j[i] \neq \perp\} \cup_{j \in [N]} \mathbf{s}.buffer[i,j] \right\}$$

$$d_{\mathcal{B}}(\mathbf{s}_1, \mathbf{s}_2) \triangleq \max_{r_1 \in U(\mathbf{s}_1), r_2 \in U(\mathbf{s}_2)} d(r_1, r_2)$$

An admissible execution α is said to converge to a untimed state $\mathbf{s}^* \in S_{\mathcal{B}}$ if $untime(\alpha(t)) \rightarrow \mathbf{s}^*$ with respect to the metric $d_{\mathcal{B}}$, as $t \rightarrow \infty$[2]. Automaton $\mathcal{B}$ converges to $\mathbf{s}^*$ if all its admissible executions converge.

4 Verification of the Partially Synchronous Systems

Throughout this section we assume that $\mathcal{A}$ is a shared state system and $\mathcal{B}$ is the corresponding partially synchronous system obtained using the translation scheme of the previous section. We denote the set of states of $\mathcal{A}$ by $S_{\mathcal{A}}$ and the individual states by s, u, etc. We assume that $\mathcal{A}$ converges to a state $s^* \in S_{\mathcal{A}}$ with respect to the metric d and a fairness condition $\mathcal{F}$. We assume that convergence of $\mathcal{A}$ is proved using Theorem 1. Therefore, we know that there exists a well ordered set $(T, <)$ with a smallest element 0 and a collection of sets $\{P_k \subseteq S |\ k \in T\}$ satisfying the conditions C1-5.

We define the following relation $\mathcal{R} \subseteq S_{\mathcal{B}} \times S_{\mathcal{A}}$:

$$\mathcal{R}(\mathbf{s}, s) \triangleq (\forall i \in [N], s_i = \mathbf{s}.x_i \ \vee \ \exists j \in [N], s_i \in \mathbf{s}.buffer[i,j] \cup \mathbf{s}.y_j[i])$$

For each i, the i-th component of s can be one of the following: (i) the state of the i-th process in $\mathbf{s}$, (ii) a message in transit from i to some j in $\mathbf{s}.buffer$, (iii) the state of the history variable $\mathbf{s}.y_j[i]$ for some other process j. If $\mathcal{R}(\mathbf{s}, s)$ then we say that s is an *asynchronous view* of $\mathbf{s}$. Given $\mathbf{s} \in S_{\mathcal{B}}$, we define $\mathcal{R}(\mathbf{s}) \triangleq \{s \in S_{\mathcal{A}} \mid \mathcal{R}(\mathbf{s}, s)\}$. We define $\mathbf{s}^* \triangleq \{\mathbf{s} \in S_{\mathcal{B}} \mid \forall s \in \mathcal{R}(\mathbf{s}) \ \ s = s^*\}$.

In the remainder of this section we shall prove that $\mathcal{B}$ converges to $\mathbf{s}^*$ with respect to the metric $d_{\mathcal{B}}$. We make the following two assumptions about the structure of the $\mathcal{P}_k$'s and message losses. For any specific problem these assumptions become proof obligations which must be discharged.

Assumption 1. *Consider any two states $s, u \in S$, a process index $i \in [N]$, and an action $a \in A_i$. For any $k, l \in T$, $l > k$, if $P_k(s)$ and $P_k(u)$ hold, then:*

B1. $P_k([s|s_i := f_{ia}(s)]) \Rightarrow P_k([u|u_i := f_{ia}(s)])$, *and*
B2. $P_l([s|s_i := f_{ia}(s)]) \Rightarrow P_l([u|u_i := f_{ia}(s)])$.

[2] $\alpha(t) \triangleq \alpha'.\mathsf{lstate}$, where α' is the longest prefix of α with $\alpha'.\mathsf{ltime} = t$.

Assumption 2. *For any $i, j \in [N]$ with i a neighbor of j, along any admissible execution α of $\mathcal{B}$, for any time t, there exists $\zeta > l + w + b$ such that j receives at least one message sent after time t from i within time $t + \zeta$.*

All processes execute send messages within w time. Hence, every every agent i receives at least one message from every neighbor in the interval $[t, t+\zeta]$. Next, we define a sequence $Q_k, k = 0, 1, 2, \ldots$ of predicates on states of $S_{\mathcal{B}}$ based on the predicates P_k on $S_{\mathcal{A}}$. Informally, Q_k holds for a state $\mathbf{s}$ exactly when all asynchronous view of $\mathbf{s}$ satisfy P_k.

$$Q_k(\mathbf{s}) \stackrel{\Delta}{=} (\forall s \in S_{\mathcal{A}}, \mathcal{R}(\mathbf{s}, s) \Rightarrow \ P_k(s)).$$

We now show that the conditions C1-5 are satisfied by the collection of sets Q_k. The proof for the next lemma uses C1-3 property of $\{P_k\}$ and appears in the full version of the paper which is available online.

Lemma 1. *The collection $\{Q_k\}$ satisfies C1-3.*

Lemma 2. *$\forall k \in T, \mathbf{s}, \mathbf{s}' \in S_{\mathcal{B}}, a \in A_{\mathcal{B}}$, if $\mathbf{s} \xrightarrow{a} \mathbf{s}'$ and $Q_k(\mathbf{s})$ then $Q_k(\mathbf{s}')$.*

Proof. Assuming $Q_k(\mathbf{s})$ holds for some $k \in T$, we show that $Q_k(\mathbf{s}')$ also holds. The proof is straightforward for $a = \mathsf{drop}$, $a = \mathsf{send}$, and for a closed trajectory of $\mathcal{B}$. Consider the case where $a = \mathsf{receive}_{ij}(m)$, $i, j \in [N]$ and $m \in X$. In order to show that $Q_k(\mathbf{s}')$, we consider any $u \in S_{\mathcal{A}}$ and assume that $\mathcal{R}(\mathbf{s}', u)$ holds. Then, it suffices to deduce $P_k(u)$.

Let the state of process i in the pre-state $\mathbf{s} \in S_{\mathcal{B}}$ be (x, y). Then its post-state is (x', y), where $x' = f_{ia}([y|y_i := x])$. We define the corresponding pre-state $s \in S_{\mathcal{A}}$ as $[y|y_i := x]$.From the definition of $\mathcal{R}$, it is follows that $\mathcal{R}(\mathbf{s}, s)$ holds. From the definition of Q and C4 we have these two implications:

$$Q_k(\mathbf{s}) \wedge \mathcal{R}(\mathbf{s}, s) \ \Rightarrow \ P_k(s) \qquad P_k(s) \Rightarrow P_k([s|s_i := f_{ia}(s)])$$

Assume that u is an asynchronous view of $\mathbf{s}'$. Then u is an asynchronous view of $\mathbf{s}$ with u_i either unchanged, or replaced by $f_{ia}(s)$. Hence:

$$\begin{aligned}
\mathcal{R}(\mathbf{s}', u) &\Rightarrow \mathcal{R}(\mathbf{s}, u) \vee (\exists v : \mathcal{R}(\mathbf{s}, v) \wedge (u = [v|v_i := f_{ia}(s)])) \\
Q_k(\mathbf{s}) \wedge \mathcal{R}(\mathbf{s}', u) &\Rightarrow (Q_k(\mathbf{s}) \wedge \mathcal{R}(\mathbf{s}, u)) \vee (\exists v : Q_k(\mathbf{s}) \wedge \mathcal{R}(\mathbf{s}, v) \wedge (u = [v|v_i := f_{ia}(s)])) \\
&\Rightarrow P_k(u) \vee (\exists v : P_k(v) \wedge (u = [v|v_i := f_{ia}(s)])) \quad [\text{From } Q \text{ definition}] \\
&\Rightarrow P_k(u) \quad [\text{from } B1, \text{ and } P_k(s)].
\end{aligned}$$

Lemma 3. *For all $k \in T$, if $P_k \neq \{s^*\}$ and $\mathbf{s} \in Q_k$ then there exists $l > k$ and a closed execution fragment α of $\mathcal{B}$ such that*

$$(untime(\alpha.\mathsf{fstate}) = \mathbf{s}) \wedge (untime(\alpha.\mathsf{lstate}) \in Q_l) \wedge (\alpha.\mathsf{ltime} \leq 2 \cdot \zeta)$$

Proof. Let us fix $k \in T$. By C5, there exists $l \in T, l > k$ and a transition function f_{ia} of $\mathcal{A}$ such that for all $s \in S_{\mathcal{A}}$ $P_k(s) \Rightarrow s' = [s \mid s_i := f_{ia}(s)] \in P_l$. We define a new relation $R' \subseteq S_{\mathcal{B}} \times S_{\mathcal{A}}$ as follows:

$$\mathcal{R}'(\mathbf{s}, s) \stackrel{\Delta}{=} \exists u, v \in S_{\mathcal{A}} : \mathcal{R}(\mathbf{s}, u) \wedge \mathcal{R}(\mathbf{s}, v) \wedge s = [v|v_i := f_{ia}(u)]$$

Thus $\mathcal{R}'(\mathbf{s}, s)$ holds exactly when s is an asynchronous view v of $\mathbf{s}$ except that the i-th agent's state is $f_{ia}(u)$ where u is itself an asynchronous view of $\mathbf{s}$. We define $Q'_k(\mathbf{s}) \triangleq (\forall u \in S_{\mathcal{A}} : \mathcal{R}'(\mathbf{s}, u) \Rightarrow P_l(u))$. If $\mathbf{s} \in Q_k \cap Q'_k$, then for all i, $\mathbf{s}.x_i$ satisfies P_l and any asynchronous view of $\mathbf{s}$ satisfies P_k.
Claim. $Q_k \cap Q'_k$ is invariant under the transitions and trajectories of $\mathcal{B}$.
Proof of Claim. The proof is straightforward for an actions drop, send and trajectories of $\mathcal{B}$. Consider an action $a = \mathsf{receive}_{j,k}(m)$. We consider two cases $i = j$ and $i \neq j$. Consider the case when $i = j$. All $\mathbf{s}.x$ satisfies P_l. From C4, P_l is invariant under transitions of $\mathcal{A}$. Hence all $\mathbf{s}'.x$ satisfy P_{k+1}. Therefore $\mathbf{s}' \in Q'_k$. Consider the case $j \neq i$. Applying Lemma 2, $\mathbf{s}' \in Q_k$ holds. Hence, for all $s' \in \mathcal{R}(s')$ we have that $s' \in P_k(s')$ and by B2, $\mathbf{s}' \in Q'_k$.

We define α as the concatenation of two fragments α_1 and α_2. We show that

1. $\exists$ a closed execution fragment α_1 with $untime(\alpha_1.\mathsf{fstate}) \in Q_k$ and $\alpha_1.\mathsf{ltime} \geq \zeta$ such that $untime(\alpha_1.\mathsf{lstate}) \in Q_k \cap Q'_k$.
2. $\forall$ closed execution fragments α_2 with $untime(\alpha_2.\mathsf{fstate}) \in Q'_k \cap Q_k$ and $\alpha_2.\mathsf{ltime} \geq \zeta$, $untime(\alpha_2.\mathsf{lstate}) \in Q_l$.

Part 1. By Assumption 2, i receives at least one message from all its neighbors by time $t + \zeta$. Denote by $\mathbf{s}' = untime(\alpha_1.lstate)$ and assume that $\mathbf{s}'$ is obtained by executing g_{ia}. By Lemma 2, $\mathbf{s}' \in Q_k$. Denote by $s' \in S_{\mathcal{A}}$ any state of $\mathcal{A}$ such that $\mathcal{R}'(\mathbf{s}', s')$. It suffices to show that $s' \in P_l$. By definition of $\mathcal{R}'$, there exists u, v such that $\mathcal{R}(\mathbf{s}', u) \wedge \mathcal{R}(\mathbf{s}', v) \wedge s' = [v \mid v_i := f_{ia}(u)]$. Since $Q_k(\mathbf{s}')$ holds, it follows that $u \in P_k$ and $v \in P_k$. By C5, $P_k(u) \Rightarrow P_l([u \mid u_i := f_{ia}(u)])$. Hence, by B2, $s' \in P_l$ with $s' = [v \mid v_i := f_{ia}(u)])$.

P*art 2.* Fix a closed execution fragment α_2 with start state $untime(\alpha_2.\mathsf{fstate}) \in Q'_k \cap Q_k$. Assume that α_2 ends at time $\alpha_2.\mathsf{ltime} \geq \zeta$. We denote $untime(\alpha_2.\mathsf{lstate})$ by $\mathbf{s}'$. We will show that $\mathbf{s}' \in Q_{k+1}$ holds. By Claim 1, $\mathbf{s}' \in Q'_k$. Let s' be any state in $\mathcal{R}(\mathbf{s}')$. It suffices to show that $s' \in P_l$ holds. By Assumption 2 (noting that $\zeta \geq b$), for all j, k $\mathbf{s}'.x$, $\mathbf{s}'.y_j[k]$, and $\mathbf{s}'.buffer[j, k]$ contain information sent at or after time 0 and this information satisfies P_l. This is because starting from time 0 the x variables satisfy P_l and by time ζ the old messages and local copies are updated with values that satisfy P_l. Hence, any asynchronous view of $\mathbf{s}'$ satisfies P_l. Hence, $P_l(s')$ holds.

Theorem 2. *If $\mathcal{A}$ converges to s^* with respect to d, then under Assumptions B1-2 and 2, $\mathcal{B}$ converges to $\mathbf{s}^*$.*

Proof. It is straightforward to see that $\mathcal{B}$ is indeed a labeled transition system with set of states $\mathcal{S}$, start states defined by the start states of $\mathcal{A}$, set of actions $A_{\mathcal{B}} \cup \mathcal{T}_{\mathcal{B}}$, and transitions $(s, a, s') \in \rightarrow$ if and only if (i) $(s, a, s') \in \mathcal{D}_{\mathcal{B}}$ or (ii) $\exists\ \tau \in \mathcal{T}_{\mathcal{B}}$, with $\tau(0) = s$ and $\tau.\mathsf{lstate} = s'$. Therefore, Theorem 1's sufficient conditions for convergence are applicable to $\mathcal{B}$ with fairness conditions replaced by time bounded progress guarantees. From Assumptions B1-2 and convergence of $\mathcal{A}$ we obtain a collection $\{Q_k\}$ of invariant sets of $\mathcal{B}$ which satisfy conditions 1-4. Assumption 2 and Lemma 3 imply that $\mathcal{B}$ makes progress.

5 Verifying Convergence of a Pattern Formation Protocol

We verify a class of pattern formation protocols for mobile agents. Starting from arbitrary locations in a space, the goal of such a protocol is to make the agents converge to some predefined spatial pattern. Distributed pattern formation protocols have been studied extensively, but typically under the assumption that the agents can communicate synchronously (see, for example [9,6,8,18]). In this paper, we present the verification of a simple one-dimensional algorithm. Several generalizations of this protocol have been presented in [7].

The shared state protocol is modeled as a LTS $\mathcal{A} = (S, S_0, A, \rightarrow)$, where (a) $S = \mathbb{R}^{N+1}$, (b) $S_0 \in \mathbb{R}^{N+1}$ (c) $A = \cup_{i\in[N]} A_i$, where $A_i \subseteq \{(i, \mathsf{avg}_{l,r}) \mid l < i < r\}$. (d) $f_{i\ \mathsf{avg}_{l,r}} : \mathbb{R}^{N+1} \rightarrow \mathbb{R}$ such that for $s \in S$, $f_{i\ \mathsf{avg}_{l,r}}(s) = \frac{r-i}{r-l} s_l + \frac{i-l}{r-l} s_r$. Note that for every $l < i$ and $r > i$, the object $(i, \mathsf{avg}_{l,r})$ may not be an action for agent i; A_i is some subset of such actions. Action $(i, \mathsf{avg}_{l,r}) \in A_i$ changes the state of the i^{th} agent according to the function $f_{i,\mathsf{avg}_{l,r}}$. This function depends on the states of agents l and r, that is $D((i, \mathsf{avg}_{l,r})) = \{l, r\}$. We adopt the notations from Section 2 to $\mathcal{A}$. For instance, for a state $s \in S$, we denote the i^{th} component as s_i. It is easy to check that $\mathcal{A}$ is a shared state system. At a particular state s of $\mathcal{A}$, we say that agent i is *located* at s_i. Throughout this section, *mid* denotes the value $\frac{N}{2}$.

We define a state $s^* \in S$ as follows: $\forall i \in [N], \quad s^*_i \triangleq s_{00}\frac{N-i}{N} + s_{0N}\frac{i}{N}$. This specifies a particular pattern where agents are located, in order, at equidistant points on a straight with extremes s_{00} and s_{0N}. We set $\mathcal{F} = \{A_i\}_{i\in[N]}$. It turns out that $\mathcal{F}$-fair executions of $\mathcal{A}$ converges to the state s^* with respect to the Euclidean metric on S. In the remainder of this section, we shall first verify this property and show how this result carries over to the convergence of the partially synchronous version of $\mathcal{A}$.

5.1 Convergence of Shared State Protocol

First, we introduce the deviation profile of a state of $\mathcal{A}$ which in turn will be used to define a sequence of predicates which satisfy C1-5. For any $x \in \mathbb{R}$ and $i \in [N]$, we define $e_i(x) \triangleq |x - s^*_i|$. Given $s \in S_{\mathcal{A}}$, $i \in [N]$, $m \in \mathbb{N}$, we define the following two symmetric predicates:

$$L_{m,j}(s) \triangleq \forall l \leq j \quad e_l(s_l) \leq C \cdot \beta^m \left(1 - \frac{1}{2^l}\right)$$

$$R_{m,j}(s) \triangleq \forall r \geq N - j \quad e_r(s_r) \leq C \cdot \beta^m \left(1 - \frac{1}{2^{N-r}}\right)$$

where $\beta \triangleq \left(1 - \frac{1}{2^N}\right)$, and C is chosen such that the $L_{0,\frac{N}{2}}$ and $R_{0,\frac{N}{2}}$ predicates are satisfied at the start state s_0. For any state s, if $L_{m,j}(s)$ holds then the deviations of the agent locations at s from those at s^* is upper-bounded by the deviation profile function (shown in Figure 3) increasing from 0 to j. Symmetrically, the predicate $R_{m,j}(s)$ holds if the deviations are decreasing from $N - j$ to N.

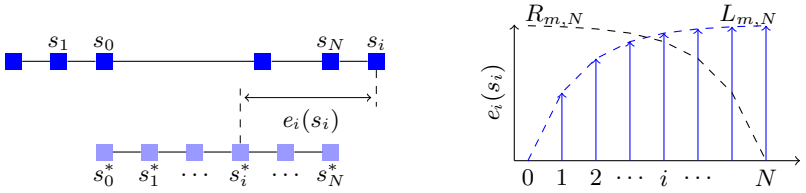

Fig. 3. Deviation from s^*. Left and Right deviation profiles.

For a state $s \in S_{\mathcal{A}}$, we define $max(s) \triangleq \max_{i\in[N]} e_i(s_i)$ and $M_m(s) \triangleq \langle max(s) \leq C \cdot \beta^m \rangle$. We define $d(s, s^*) \triangleq max(s)$. For any $j \in [mid]$, we define the following symmetric predicates: $\mathcal{L}_{m,j}(s) \triangleq M_m(s) \wedge L_{m,j}(s) \wedge L_{m-1,mid}(s)$ and $\mathcal{R}_{m,j}(s) \triangleq M_m(s) \wedge R_{m,j}(s) \wedge R_{m-1,mid}(s)$.

These predicates partition $[N]$ into three groups as: for all $i \in [j]$, the deviation for agent i is upper bounded by the profile function defined by $C \cdot \beta^m$; for $i \in \{j+1 \ldots mid\}$ the upper bound is $C \cdot \beta^{m-1}$; and for the remaining, the deviations do not have any upper bound other than one given by the first part of the predicate ($\leq C \cdot \beta^m$). For $i \in [N]$, we define the left profile function $lp_{m,j}(i)$ as $C \cdot \beta^m \left(1 - \frac{1}{2^i}\right)$ if $i \leq j$, equals to $C \cdot \beta^{m-1} \left(1 - \frac{1}{2^i}\right)$ if $j < i \leq mid$ and $C \cdot \beta^m$ otherwise. This function is concave.

Lemma 4. *Let T be the set $\mathbb{N} \times [mid]$ equipped with lexicographic ordering ($\leq_{lex}$). The collections $\{\mathcal{L}_{m,j}\}$ and $\{\mathcal{R}_{m,j}\}$ indexed by T satisfy C1-4.*

Proof. C1. Consider any state $s \in \mathcal{L}_{m_2,j_2}$ and any pair $[m_1, j_1] \leq_{lex} [m_2, j_2]$. If $m = m_1 = m_2$, $s \in \mathcal{L}_{m,j_2} \Rightarrow s \in \mathcal{L}_{m,j_1}$ since the profile holds up to j_2, it is valid up to j_1 (for all $j_1 \leq j_2$). When $m_1 < m_2$, $s \in \mathcal{L}_{m_2,j_2} \Rightarrow s \in \mathcal{L}_{m_1,j_1}$ for all $j_1, j_2 \leq mid$; this is because for all i $lp_{m_2,j_2}(i) \leq lp_{m_1,j_1}(i)$ since $\beta^{m_2} < \beta^{m_2-1} \leq \beta^{m_1}$. For C2, for all ϵ we set k to be any value satisfying $C \cdot \beta^k \leq \epsilon$. Hence, $\forall s$ satisfying $\mathcal{L}_{k,0}$ we have that $max(s) \leq C \cdot \beta^k < \epsilon$. C3 follows from the definition of C. C4. Assume without loss of generality $s \in \mathcal{L}_{m,j}$ and $a = (i, \mathsf{avg}_{l,r})$. For all $j \neq i$, s'_j satisfies $\mathcal{L}_{m,j}$, since $s'_j = s_j$. Assume $i \leq j$. The value s'_i satisfies $\mathcal{L}_{m,j}$ as well, and $e_i(s'_i)$ is upper bounded by

$$\frac{r-i}{r-l} e_l(s_l) + \frac{i-l}{r-l} e_r(s_r) \leq \frac{r-i}{r-l}\left(1 - \frac{1}{2^l}\right) C \cdot \beta^m + \frac{i-l}{r-l} C \cdot \beta^m \leq C \cdot \beta^m \left(1 - \frac{1}{2^i}\right)$$

An analogous argument is used to prove the case when $i > j$.

Condition C5 is only partially satisfied by these predicates; for any m and $j < mid$, for all $\mathcal{L}_{m,j}$ (resp. $\mathcal{R}_{m,j}$) there exists an action such that the execution of this action take the system to $\mathcal{L}_{m,j+1}$ (resp. $\mathcal{R}_{m,j+1}$). The following relationships among $\mathcal{L}$ and $\mathcal{R}$ are used for showing C5. The proofs appears in the full version.

Lemma 5. $\forall m \in \mathbb{N}, \quad \mathcal{L}_{m,mid} \cap \mathcal{R}_{m,mid} = \mathcal{L}_{m+1,0} \cap \mathcal{R}_{m+1,0}$

Lemma 6. *$\forall j < mid$ (a) $\exists a_1$ such that $\forall s \xrightarrow{a_1} s'$ and $\forall m \in \mathbb{N}$, $s \in \mathcal{L}_{m,j} \Rightarrow s' \in \mathcal{L}_{m,j+1}$. (b) $\exists a_2$ such that $\forall s \xrightarrow{a_2} s'$ and $\forall m \in \mathbb{N}$, $s \in \mathcal{R}_{m,j} \Rightarrow s' \in \mathcal{R}_{m,j+1}$.*

Lemma 5 implies that the left and right profile predicates satisfy C1-4, but in order to prove C5 we require both these predicates hold simultaneously. This motivates our next definition. For state $s \in S_{\mathcal{A}}$, $m \in \mathbb{N}$, $j \in [mid-1]$, $b \in \{0,1\}$, we define: $\mathcal{P}_{m,j,b}(s) \triangleq \mathcal{L}_{m,j+b}(s) \wedge \mathcal{R}_{m,j}(s)$. All indices from 0 to $j+b$ and from $N-j$ to N belong to the profile defined by $C \cdot \beta^m$, while the indices between $(j+b)+1$ and $(N-j)-1$ belong to profile defined by $C \cdot \beta^{m-1}$.

Lemma 7. *Let T be the set $\mathbb{N} \times [mid-1] \times \{0,1\}$ equipped with lexicographic ordering ($\leq_{lex}$). The collection $\{\mathcal{P}_{m,j,b}\}$ indexed by T satisfies C1-5.*

Proof. It is straightforward to check using Lemma 4 that the sequence of predicates satisfy C1-4. C5. Applying Part (a) of Lemma 6,

$$s \in \mathcal{P}_{m,j,0} \Rightarrow s \in \mathcal{L}_{m,j} \wedge s \in \mathcal{R}_{m,j} \Rightarrow s' \in \mathcal{L}_{m,j+1} \wedge s' \in \mathcal{R}_{m,j} \Leftrightarrow s' \in \mathcal{P}_{m,j,1}.$$

for any m, j with $j \leq mid-1$, let a_1 be any action in A_{j+1}. Without loss of generality, we assume $a_1 = (j+1, \mathsf{avg}_{l,r})$. Using part (b) of Lemma 6, we obtain

$$s \in \mathcal{P}_{m,j,1} \Rightarrow s \in \mathcal{L}_{m,j+1} \wedge s \in \mathcal{R}_{m,j} \Rightarrow s' \in \mathcal{L}_{m,j+1} \wedge s' \in \mathcal{R}_{m,j+1} \Leftrightarrow s' \in \mathcal{P}_{m,j+1,0}.$$

Next, for any m, j with $j < mid-1$, let a_2 be any action in $A_{N-(j+1)}$. Again, without loss of generality, let $a_2 = (N-(j+1), \mathsf{avg}_{l,r})$. Finally from Lemma 5, $s \in \mathcal{P}_{m,mid-1,1} \Rightarrow s \in \mathcal{P}_{m+1,0,0}$. Since both A_{j+1} and $A_{N-(j+1)}$ are in the fairness condition $\mathcal{F}$, we obtain the required result.

Lemma 7 and Theorem 1 imply that all $\mathcal{F}$-fair executions of $\mathcal{A}$ converge to s^*.

5.2 Convergence of the Partially Synchronous Protocol

From the shared state protocol for patten formation described in Section 5.1, we first obtain the corresponding $\mathsf{Process}_i$ automaton based on the translation scheme of Section 3. In particular, $\mathsf{Process}_i$ is a TIOA specified by the code in Figure 1 with $X = \mathbb{R}$, $x_0 = s_{0i}$ and $g_{i\ avg_{l,r}} : \mathbb{R}^3 \to \mathbb{R}$. The $g_{i\ avg_{l,r}}$ functions are obtained from the $f_{i\ avg_{l,r}}$ functions using the transformation of Equation 1. The communication channel for the system is modeled by LBCast of Figure 2 with $X = \mathbb{R}$ and some value for b. The complete partially synchronous system specification is the TIOA obtained by composing $\mathsf{Process}_i$'s with LBCast. Finally, the convergence state $\mathbf{s}^*$ and $d_{\mathcal{B}}$ for $\mathcal{B}$ are obtained from s^*, d of $\mathcal{A}$ using the definitions in 3. It is easily checked that the collection of predicates $\{\mathcal{P}_{m,j,b}\}$ satisfy Assumptions 1 and 2. Therefore, from Theorem 2, we conclude that $\mathcal{B}$ converges to $\mathbf{s}^*$. In fact, we observe that the system $\mathcal{B}$ converges under the following weaker assumption about message losses:

Assumption 3. *For any agent i, for any time t there exists $\zeta > 0$, such that i receives at least one message sent after time t from some agent $l < i$ ($r > i$, respectively) within time $t + u$ with $(i, \mathsf{avg}_{l,r}) \in A_i$.*

This is weaker than Assumption 2 since each process i receives at least one message from some pair (neighbor) and not necessarily all pairs in A_i. The progress property is still guaranteed because by the system makes progress executing any action of A_i.

5.3 Verification in PVS Theorem Prover

We have developed a PVS [19] theory for verifying partially synchronous pattern formation protocols within the exiting Timed I/O Automata/PVS framework [3,15]. The theory formalizes partially synchronous systems as described in this paper, and we have verified the convergence of the example presented here. The PVS theory files and the related documentation are available from `http://www.infospheres.caltech.edu/papers`. The the proofs presented in this section have been mechanically checked using the PVS theorem prover. The invariance of the $\mathcal{P}$ predicates are proved using the standard inductive proof technique followed by a case analysis on the actions (and trajectories) of the automaton in question. We also prove the convergence of the partially synchronous system directly under Assumption 3. An appropriately changed version of Lemma 5 holds in the partially synchronous settings as well. In order to do so, we prove a set of basic lemmas about LBCast that are used repeatedly. One example, of such a basic lemma is that if all the input messages satisfy a certain predicate, then within bounded time the values stored in the *buffer* satisfy the same predicate

6 Discussion

Designing and verifying partially synchronous distributed algorithms is complicated because of their inherent concurrency and message delays. We have presented a methodology for transforming a shared state distributed system—in which processes can read each other's state without delay—to a partially synchronous system, such that the convergence of the former carry over to the latter, under certain assumptions. Checking Assumption 1 is easy when it can be expressed as a conjunction of predicates on individual process states. It would be interesting to explore relaxations of this assumption. Assumption 2 is fairly weak, however, it is possible to weaken it further for specific protocols—as it is observed in the presented case study. We implemented the theory in PVS and have applied this methodology to verify the convergence of a mobile-agent pattern pattern formation protocol operating on partially synchronous communication. Several generalizations of the translation scheme and the convergence theorem are possible; some more immediate than others. The processes participating in the partially synchronous system could have clocks with bounded drift. We could also define arbitrary continuous trajectories for the main state variables x_i as long as Assumption 1 is satisfied.

References

1. Tempo toolset, version 0.2.2 beta, January 2008. `http://www.veromodo.com/`.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. M. Archer, C. Heitmeyer, and S. Sims. TAME: A PVS interface to simplify proofs for automata models. In *Proceedings of UITP '98*, July 1998.
4. M. Archer, H. Lim, N. Lynch, S. Mitra, and S. Umeno. Specifying and proving properties of timed I/O automata using Tempo. *Design Aut. for Emb. Sys.*, 2008. To appear.
5. J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL in 1995. In *TACAS '96*, pages 431–434, 1996.
6. V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis. Convergence in multiagent coordination consensus and flocking. In *CDC-ECC*, pages 2996–3000, 2005.
7. —, Formations of mobile agents with message loss and delay, 2007. preprint `http://www.ist.caltech.edu/~mitras/research/2008/asynchcoord.pdf`.
8. S. Chatterjee and E. Seneta. Towards consensus: some convergence theorems on repeated averaging. *J. Applied Probability*, 14(1):89–97, 1977.
9. S. Clavaski, M. Chaves, R. Day, P. Nag, A. Williams, and W. Zhang. Vehicle networks: achieving regular formation. In *ACC*, 2003.
10. C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *JACM*, 35(2):288–323, 1988.
11. M. Hendriks. Model checking the time to reach agreement. In *FORMATS '05*, pages 98–111, November 2005.
12. D. Kaynar, N. Lynch, S. Mitra, and S. Garland. *TIOA Language*. MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, 2005.
13. D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on CS. Morgan Claypool, November 2005.
14. L. Lamport. Real-time model checking is really simple. In *CHARME '05*, *LNCS* 3725, 162–175. Springer, 2005.
15. H. Lim, D. Kaynar, N. Lynch, and S. Mitra. Translating timed I/O automata specifications for theorem proving in PVS. In *FORMATS'05*, LNCS 3829, 17–31 September 2005.
16. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., 1996.
17. S. Mitra and K. M. Chandy. A formalized theory for verifying stability and convergence of automata in pvs. To appear in *TPHOLs '08*.
18. R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. In *Proc. of the IEEE*, 95(1):215–233, January 2007.
19. S. Owre, S. Rajan, J. Rushby, N. Shankar, and M. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *CAV '96*, LNCS 1102, 411–414, July/August 1996.
20. J. N. Tsitsiklis. On the stability of asynchronous iterative processes. *Theory of Computing Systems*, 20(1):137–153, December 1987.