On Asynchrony, Memory, and Communication: Separations and Landscapes

Paola Flocchini
EECS, University of Ottawa, Canada

Nicola Santoro
School of Computer Science, Carleton University, Ottawa, Canada

Yuichi Sudo
Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan

Koichi Wada
Faculty of Science and Engineering, Hosei University, Tokyo, Japan

Abstract

Research on distributed computing by a team of identical mobile computational entities, called robots, operating in a Euclidean space in *Look-Compute-Move* (*LCM*) cycles, has recently focused on better understanding how the computational power of robots depends on the interplay between their internal capabilities (i.e., persistent memory, communication), captured by the four standard computational models ($\mathcal{OBLOT}$, $\mathcal{LUMI}$, $\mathcal{FSTA}$, and $\mathcal{FCOM}$) and the conditions imposed by the external environment, controlling the activation of the robots and their synchronization of their activities, perceived and modeled as an adversarial scheduler.

We consider a set of adversarial asynchronous schedulers ranging from the classical *semi-synchronous* (SSYNCH) and *fully asynchronous* (ASYNCH) settings, including schedulers (emerging when studying the atomicity of the combination of operations in the *LCM* cycles) whose adversarial power is in between those two. We ask the question: what is the computational relationship between a model M_1 under adversarial scheduler K_1 ($M_1(K_1)$) and a model M_2 under scheduler K_2 ($M_2(K_2)$)? For example, are the robots in $M_1(K_1)$ more powerful (i.e., they can solve more problems) than those in $M_2(K_2)$?

We answer all these questions by providing, through cross-model analysis, a complete characterization of the computational relationship between the power of the four models of robots under the considered asynchronous schedulers. In this process, we also provide qualified answers to several open questions, including the outstanding one on the proper dominance of SSYNCH over ASYNCH in the case of unrestricted visibility.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Look-Compute-Move, Oblivious mobile robots, Robots with lights, Memory versus Communication, Moving and Computing, Asynchrony

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2023.28

Related Version *Full Version*: `https://arxiv.org/abs/2311.03328`

Funding This research was partly supported by NSERC through the Discovery Grant program, by JSPS KAKENHI No. 20H04140, 20KK0232, 20K11685, 21K11748, and by JST FOREST Program JPMJFR226U.

1 Introduction

1.1 Background

Robot Models. Since the seminal work of Suzuki and Yamashita [31], the studies of the computational issues arising in distributed systems of mobile computational entities, called *robots*, operating in a Euclidean space have focused on identifying the minimal assumptions

27th International Conference on Principles of Distributed Systems (OPODIS 2023).
Editors: Alysson Bessani, Xavier Défago, Junya Nakamura, Koichi Wada, and Yukiko Yamauchi; Article No. 28; pp. 28:1–28:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

on *internal* capabilities of the robots (e.g., persistent memory, communication) and *external* conditions of the system (e.g., synchrony, activation scheduler) that allow the entities to perform basic tasks and collectively solve given problems.

Endowed with computational, visibility and motorial capabilities, the robots are anonymous (*i.e.*, indistinguishable from each other), uniform (*i.e.*, run the same algorithm), and disoriented (*i.e.*, they might not agree on a common coordinate system). Modeled as mathematical points in the 2D Euclidean plane in which they can freely move, they operate in *Look-Compute-Move* (*LCM*) cycles. In each cycle, a robot "*Look*s " at its surroundings obtaining (in its current local coordinate system) a snapshot indicating the locations of the other robots. Based on this information, the robot executes its algorithm to "*Compute*" a destination, and then "*Move*s" towards the computed location.

In the (weakest and de facto) standard model, $\mathcal{OBLOT}$, the robots are also oblivious (*i.e.*, they have no persistent memory of the past) and silent (*i.e.*, they have no explicit means of communication). Extensive investigations have been carried out to understand the computational limitations and powers of $\mathcal{OBLOT}$ robots for basic coordination tasks such as Gathering (e.g., [1, 2, 4, 8, 9, 10, 17, 25, 31]), Pattern Formation (e.g., [18, 22, 31, 34, 35]), Flocking (e.g., [7, 23, 30]); see also the monograph [14] for a general account.

The absence of persistent memory and the lack of explicit communication critically restrict the computational capabilities of the $\mathcal{OBLOT}$ robots, and limit the solvability of problems. These limitations are removed, to some extent, in the $\mathcal{LUMI}$ model of *luminous* robots. In this model, each robot is equipped with a constant-bounded amount of persistent[1] memory, called *light*, whose value, called *color*, is visible to all robots. In other words, luminous robots can both remember and communicate, albeit at a very limited level. Since its introduction in [11], the model has been the subject of several investigations focusing on the design of algorithms and the feasibility of problems for $\mathcal{LUMI}$ robots (e.g. [3, 11, 12, 20, 24, 27, 28, 29, 32, 33]; see Chapter 11 of [14] for a recent survey). An important result is that, even if so limited, the simultaneous presence of both persistent memory and communication renders luminous robots strictly more powerful than oblivious robots [11]. This has in turns opened the question on the individual computational power of the two internal capabilities, memory and communication, and motivated the investigations on two sub-models of $\mathcal{LUMI}$: the *finite-state* robots denoted as $\mathcal{FSTA}$, where the robots have a constant-size persistent memory but are silent, and the *finite-communication* robots denoted as $\mathcal{FCOM}$, where robots can communicate a constant number of bits but are oblivious (e.g., see [5, 6, 20, 21, 27, 28]).

A/Synchrony. All these studies in all those models have brought to light the crucial role played by two interrelated *external* factors: the level of synchronization and the activation schedule provided by the system. Like in other types of distributed computing systems, there are two different settings, the synchronous and the asynchronous ones.

In the *synchronous* (also called *semi-synchronous*) (SSYNCH) setting, introduced in [31], time is divided into discrete intervals, called *rounds*. In each round, an arbitrary but nonempty subset of the robots is activated, and they simultaneously perform exactly one *Look-Comp-Move* cycle. The selection of which robots are activated at a given round is made by an adversarial scheduler, constrained only to be fair, i.e., every robot is activated infinitely often. Weaker synchronous adversaries have also been introduced and investigated. The most important and extensively studied is the *fully-synchronous* (FSYNCH) scheduler,

[1] i.e., it is not automatically reset at the end of a cycle.

which activates all the robots in every round. Other interesting synchronous schedulers are RSYNCH, where the sets of robots activated in any two consecutive rounds are restricted to be disjoint, and it studied for its use to model energy-restricted robots [6], as well as the family of *sequential* schedulers (e.g., ROUNDROBIN), where in each round only one robot is activated.

In the *asynchronous* setting (ASYNCH), introduced in [16], there is no common notion of time, each robot is activated independently of the others; it allows for finite but arbitrary delays between the *Look*, *Comp* and *Move* phases, and each movement may take a finite but arbitrary amount of time. The duration of each cycle of a robot, as well as the decision of when a robot is activated, are controlled by an adversarial scheduler, constrained only to be fair, i.e., every robot must be activated infinitely often.

Weaker adversaries are easily identified considering the atomicity of the combination of the *Look*, *Comp* and *Move* stages. In particular, if in every cycle the three operations are executed as a single atomic instantaneous operation, this scheduler we shall call *LCM*-**atomic**-ASYNCH coincides with SSYNCH. On the other hand, by combining fewer operations, two asynchronous schedulers are identified [27]: *LC*-**atomic**-ASYNCH, where the *Look* and *Comp* operations are a single atomic operation; and *CM*-**atomic**-ASYNCH, where the *Comp* and *Move* operations are a single atomic operation.

Of independent interest is the restricted asynchronous adversary unable to schedule the *Look* operation of a robot during the *Move* operation of another. The particular theoretical relevance of this scheduler, called *M*-**atomic**-ASYNCH [27] derives from the fact that one of the strongest debilitating effects of unrestricted asynchrony is precisely the fact that a robot, when looking, cannot detect if another robot is still or moving.

Separators. Like in other types of distributed systems, understanding the computational difference between (levels of) synchrony and asynchrony has been a primary research focus, first in the $\mathcal{OBLOT}$ model, and subsequently in the others.

Indeed, one of the first results in the field has been the proof that in $\mathcal{OBLOT}$ the simple problem of two robots meeting at the same location, called *Rendezvous*(RDV), is unsolvable under SSYNCH [31] while easily solvable under FSYNCH, implying that fully synchronous $\mathcal{OBLOT}$ robots are strictly more powerful than semi-synchronous ones.

Any problem that, like *Rendezvous*, proves the separation between the computational power of robots in two different settings is said to be a *separator*. The quest has immediately been to determine if there are other problems in $\mathcal{OBLOT}$ separating SSYNCH from FSYNCH (i.e., the extent of their computational difference); no other has been found so far. Clearly more important and pressing has been the question of whether there is any computational difference between synchrony and asynchrony. The quest for a problem separating ASYNCH from SSYNCH has been ongoing for more than two decades. Recently a separator has been found in the special case when the visibility range of the robots is limited [26], leaving the existence of a separator open for the unrestricted case.

The quest for a separator in $\mathcal{OBLOT}$ has been made more pressing since the result that no separation exists between ASYNCH and SSYNCH in the $\mathcal{LUMI}$ model [11]; that is, the presence of a limited form of communication and memory is sufficient to completely overcome the limitations imposed by asynchrony. This result has motivated the investigation of the two submodels of $\mathcal{LUMI}$ where the robots are endowed with only the limited form of persistent memory, $\mathcal{FSTA}$, or of communication, $\mathcal{FCOM}$. While separation between fully synchrony and semi-synchrony has been shown to exist for both submodels [5, 21], the more important question of whether one of them is capable of overcoming asynchrony has not yet been answered; indeed, no separator between SSYNCH and ASYNCH has been found so far for either submodel.

Landscapes. To understand the impact that the factors of persistent memory and communication have on the feasibility of problems, the main investigation tool has been the comparative analysis of the (new and/or existing) results obtained for the same problems under the different four models $\mathcal{OBLOT}, \mathcal{FSTA}, \mathcal{FCOM}, \mathcal{LUMI}$. The same methodological tool can obviously be used also to establish the computational relationships between those models within a spectrum of schedulers, so to identify the relative powers of those schedulers within each model.

Through this type of cross-model analysis, researchers have recently produced a comprehensive characterization of the computational relationship between the four models with respect to the range of synchronous schedulers <FSYNCH, RSYNCH, SSYNCH>. creating a comprehensive map of the *synchronous landscape* for distributed systems of autonomous mobile robots in the four models [5, 21].

With respect to the (more powerful) *asynchronous* adversarial schedulers, ranging from LCM-**atomic**-ASYNCH (i.e., SSYNCH) to ASYNCH, very little is known to date on the computational power of persistent memory and of explicit communication in general, and on the computational relationship between the four models in particular. As mentioned, it is known that in $\mathcal{LUMI}$, robots have in ASYNCH the same computational power as in SSYNCH and that asynchronous luminous robots are strictly more powerful than oblivious synchronous robots [11].

Summarizing, while a comprehensive computational map has existed for the synchronous landscape, only disconnected fragments exist so far of the *asynchronous landscape*.

1.2 Contributions

In this paper, we analyze the computational relationship among the four models $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$ and $\mathcal{LUMI}$, under the range of asynchronous schedulers < LCM-**atomic**-ASYNCH, LC-**atomic**-ASYNCH, CM-**atomic**-ASYNCH, M-**atomic**-ASYNCH, and ASYNCH>, establishing a large variety of results. Through these results, we close several open problems, and create a complete map of the asynchronous landscape for distributed systems of autonomous mobile robots in the four models.

Among our contributions, we prove the existence of a separator between SSYNCH and ASYNCH in the standard $\mathcal{OBLOT}$ model for the unrestricted visibility case by identifying a simple natural problem, `Monotone Line Convergence` (`MLCv`), that separates SSYNCH from ASYNCH for $\mathcal{OBLOT}$ robots. This problem requires two robots to convergence towards each other monotonically (i.e., without ever increasing their distance) on the line connecting them. We prove that this problem, trivially solvable in semi-synchronous systems, is however unsolvable if the system is asynchronous.

Because of this separation in $\mathcal{OBLOT}$ on one hand, and of the known absence of separation in $\mathcal{LUMI}$ on the other, the next immediate question is whether either of $\mathcal{LUMI}$'s specific features (i.e., constant-sized communication and persistent memory) is strong enough alone to overcome asynchrony. In other words, are there separators between SSYNCH and ASYNCH in $\mathcal{FSTA}$? in $\mathcal{FCOM}$? In these regards, we provide a positive answer to both questions, thus proving that both features are needed to overcome asynchrony.

The characterization of the computational relationship between the four models with respect to the range of asynchronous schedulers is *complete*: for any two models, $M_1, M_2 \in \{\mathcal{OBLOT}, \mathcal{FSTA}, \mathcal{FCOM}, \mathcal{LUMI}\}$ and adversarial schedulers $K_1, K_2 \in \{LCM$-**atomic**-ASYNCH, LC-**atomic**-ASYNCH, CM-**atomic**-ASYNCH, M-**atomic**-ASYNCH, ASYNCH$\}$ it is determined whether the computational power of (the robots in) M_1 under K_1 is stronger than, weaker than, equivalent to or orthogonal to (i.e., incomparable with) that of (the robots in) M_2 under K_2.

For example, we prove that for $\mathcal{FSTA}$ (i.e., in presence of only limited internal persistent memory), SSYNCH is computationally more powerful than MOVE-**atomic**-ASYNCH, which in turn is computationally more powerful than ASYNCH. The several orthogonality (i.e., incomparability) results include for example the fact that the combination of asynchrony and limited persistent memory is neither more nor less powerful than the combination of synchrony and obliviousness. Observe that to prove that a model under a specific scheduler is stronger than or orthogonal to another model and scheduler (or same model and a different scheduler, or other model and same scheduler) requires to determine a problem solvable in one setting but not in the other.

Among the equivalence of two models each under a specific scheduler, we have proved that for $\mathcal{FCOM}$ (i.e., in presence of only limited communication): the atomic combination of *Compute* and *Move* does not provide any gain with respect to complete asynchrony; on the other hand, the atomic combination of *Look* and *Compute* completely overcomes asynchrony. The proof of the equivalence has involved designing a *simulation protocol* that allows to correctly execute any protocol for the first model and scheduler into the other model and scheduler.

The resulting asynchronous landscape is shown in Figure 1 where S, A, A_{LC} A_M, and A_{CM} denote SSYNCH, ASYNCH, LC-**atomic**-ASYNCH, M-**atomic**-ASYNCH, and CM-**atomic**-ASYNCH, respectively; a box located higher than another indicates dominance unless they are connected by a dashed line, which denotes orthogonality; equivalence is indicated directly in the boxes.

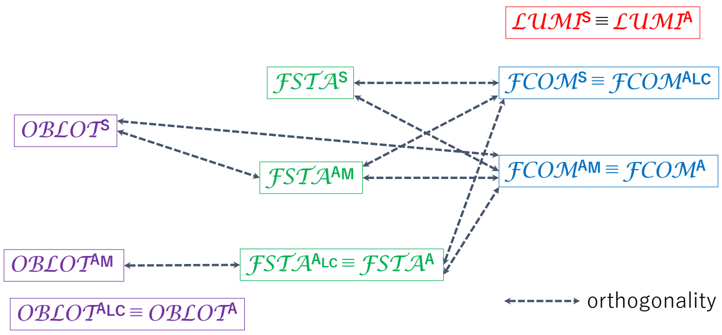

Figure 1 Asynchronous landscape of $\mathcal{LUMI}$, $\mathcal{FCOM}$, $\mathcal{FSTA}$ and $\mathcal{OBLOT}$.

Due to space limitations, some proofs and detailed descriptions are omitted; they can be found in [19].

2 Models and Preliminaries

2.1 Robots

We shall consider a set $R = \{r_0, \cdots, r_{n-1}\}$ of $n > 1$ mobile computational entities, called *robots*, operating in the Euclidean plane $\mathbb{R}^2$. The robots are *anonymous* (i.e., they are indistinguishable by their appearance), *autonomous* (i.e., without central control), *homogeneous* (i.e., the all execute the same program). Viewed as points they can move freely in the plane. Each robot is equipped with a local coordinate system (in which it it is always at its origin), and it is able to observe the positions of the other robots in its local coordinate system. The robots are *disoriented*; that is, there might not be consistency between the coordinate systems of different robots at the same time, or the same robot at different times[2]. We assume that the robots however have *chirality*; that is, they agree on the the same circular orientation of the plane (e.g., "clockwise" direction).

At any time, a robot is either *active* or *inactive*. When active, a robot r executes a *Look-Compute-Move* (*LCM*) cycle. Each cycle is compose of three operations:

1. *Look:* The robot obtains an instantaneous snapshot of the positions occupied by the other robots (expressed in its own coordinate system)[3]. We do not assume that the robots are capable of strong multiplicity detection [15].
2. *Compute:* The robot executes its algorithm using the snapshot as input. The result of the computation is a destination point.
3. *Move:* The robot moves to the computed destination[4]; if the destination is the current location, the robot stays still and the move is said to be null.

After executing a cycle, a robot becomes inactive. All robots are initially inactive. The time it takes to complete a cycle is assumed to be finite and the operations *Look* and *Compute* are assumed to be instantaneous.

In the standard model, $\mathcal{OBLOT}$, the robots are also *silent*: they have no explicit means of communication; furthermore, they are *oblivious*: at the start of a cycle, a robot has no memory of observations and computations performed in previous cycles.

In the other common model, $\mathcal{LUMI}$, each robot r is equipped with a persistent register $Light[r]$, called *light*, whose value called *color*, is from a constant-sized set C and is visible by the robots. The color of the light can be set in each cycle by r at the end of its *Compute* operation, and is not automatically reset at the end of a cycle. In $\mathcal{LUMI}$, the *Look* operation produces a colored snapshot; i.e., it returns the set of pairs $(position, color)$ of the other robots. It is sometimes convenient to describe a robot r as having $k \geq 1$ lights, denoted $r.light_1, \ldots, r.light_k$, where the values of $r.light_i$ are from a finite set of colors C_i, and to consider $Light[r]$ as a k-tuple of variables; clearly, this corresponds to r having a single light that uses $\Pi_{i=1}^{k}|C_i|$ colors. Note that if $|C| = 1$, this case corresponds to the $\mathcal{OBLOT}$ model.

Two submodels of $\mathcal{LUMI}$, $\mathcal{FSTA}$ and $\mathcal{FCOM}$, have been defined and investigated, each offering only one of its two capabilities, persistent memory and direct means of communication, respectively. In $\mathcal{FSTA}$, a robot can only see the color of its own light; thus, the color merely encodes an internal state. Therefore, robots are *silent*, as in $\mathcal{OBLOT}$, but they are *finite-state*. In $\mathcal{FCOM}$, a robot can only see the color of the light of the other robots; thus, a robot can communicate to the other robots the color of its light but does not remember its own state (color). Thus, robots are enabled with *finite-communication* but are *oblivious*.

In all the above models, a *configuration* $\mathcal{C}(T)$ at time T is the multiset of the n pairs $(r_i(T), c_i(T))$, where $c_i(T)$ is the color of robot r_i at time T.

---

[2] This is also called *variable* disorientation; restricted forms (e.g., *static* disorientation, where each local coordinate system remains always the same) have been considered for these systems.

[3] This is called the *full visibility* (or unlimited visibility) setting; restricted forms of visibility have also been considered for these systems [17].

[4] This is called the *rigid mobility* setting; restricted forms of mobility (e.g., when movement may be interrupted by an adversary), called *non-rigid mobility* have also been considered for these systems.

2.2 Schedulers, Events

With respect to the activation schedule of the robots, and the duration of their *LCM* cycles, the fundamental distinction is between the *synchronous* and *asynchronous* settings.

In the *synchronous* setting (SSYNCH), also called *semi-synchronous* and first studied in [31], time is divided into discrete intervals, called *rounds*; in each round, a non-empty set of robots is activated and they simultaneously perform a single *Look*-*Comp*-*Move* cycle in perfect synchronization. The selection of which robots are activated at a given round is made by an adversarial scheduler, constrained only to be fair (i.e., every robot is activated infinitely often). The particular synchronous setting, where every robot is activated in every round is called *fully-synchronous* (FSYNCH). In a synchronous setting, without loss of generality, the expressions "i-th round" and "time $t = i$" are used as synonyms.

In the *asynchronous* setting (ASYNCH), first studied in [16], there is no common notion of time, the duration of each phase is finite but unpredictable and might be different in different cycles, and each robot is activated independently of the others. The duration of the phases of each cycle as well as the decision of when a robot is activated is controlled by an adversarial scheduler, constrained only to be fair, i.e., every robot must be activated infinitely often.

In the asynchronous settings, the execution by a robot of any of the operations *Look*, *Compute* and *Move* is called an *event*. We associate relevant time information to events: for the *Look* (resp., *Compute*) operation, which is instantaneous, the relevant time is T_L (resp., T_C) when the event occurs; for the *Move* operation, these are the times T_B and T_E when the event begins and ends, respectively. Let $\mathcal{T} = \{T_1, T_2, ...\}$ denote the infinite ordered set of all relevant times; i.e., $T_i < T_{i+1}, i \in \mathbb{N}$. In the following, to simplify the presentation and without any loss of generality, we will refer to $T_i \in \mathcal{T}$ simply by its index i; i.e., the expression "time t" will be used to mean "time T_t".

In our analysis of ASYNCH, we will also consider and make use of the following submodels of ASYNCH, defined by the level of atomicity of the *Look*, *Comp* and *Move* operations.

- ***LC*-atomic-**ASYNCH: The scheduler does not allow any robot r to perform a *Look* operation while another robot $r' \neq r$ is performing its *Comp* operation in that cycle [13, 27]. Thus, in the *LC*-atomic-ASYNCH model, it can be assumed that, in every cycle, the *Look* and *Comp* operations are performed simultaneously and atomically and that $t_L = t_C$.
- ***M*-atomic-**ASYNCH: The scheduler does not allow any robot r to perform a *Look* operation while another robot $r' \neq r$ is performing its *Move* operation in that cycle [13, 27]. In this case, *Move* operations (called *M*-operations) in all cycles can be considered to be performed instantaneously and that $t_B = t_E$.
- ***CM*-atomic-**ASYNCH: The scheduler does not allow any robot r to perform a *Look* operation while another robot $r' \neq r$ is performing a *Comp* or *Move* operation in that cycle. Thus, in this model, in every cycle the operations *Comp* and *Move*, denoted as CPM, can be considered as performed simultaneously and atomically, and $t_C = t_B = t_E$.

To complete the description, two additional specifications are necessary.
Specification 1. In presence of visible external lights (i.e., models $\mathcal{LUMI}$ and $\mathcal{FCOM}$), if a robot r changes its color in the *Comp* operation at time $t \in \mathcal{T}$, by definition, its new color will become visible only at time $t + 1$.

Specification 2. Under the ***M*-atomic-**ASYNCH and ***CM*-atomic-**ASYNCH schedulers, if a robot r ends a non-null *Move* operation at time $t \in \mathcal{T}$, by definition, its new position will become visible only at time $t + 1$.

Note that, the model where the *Look*, *Comp*, and *Move* operations are considered as a single instantaneous atomic operation (thus referable to as ***LCM*-atomic-**Asynch is obviously equivalent to Ssynch.

In the following, for simplicity of notation, we shall use the symbols F, S, A, A_{LC}, A_M, and A_{CM} to denote the schedulers Fsynch, Ssynch, Asynch, *LC*-**atomic**-Asynch, *M*-**atomic**-Asynch, and *CM*-**atomic**-Asynch, respectively.

2.3 Problems and Computational Relationships

Let $\mathcal{M} = \{\mathcal{LUMI}, \mathcal{FCOM}, \mathcal{FSTA}, \mathcal{OBLOT}\}$ be the set of models under investigation and $\mathcal{S} = \{F, S, A, A_{LC}, A_M, A_{CM}\}$ be the set of schedulers under consideration.

A problem to be solved (or task to be performed) is described by a set of *temporal geometric predicates*, which implicitly define the *valid* initial, intermediate, and (if existing) terminal[5] configurations, as well as restrictions (if any) on the size n of the set R of robots.

An algorithm $\mathcal{A}$ *solves* a problem P in model $M \in \mathcal{M}$ under scheduler $K \in \mathcal{S}$ if, starting from any valid initial configuration, any execution by R of $\mathcal{A}$ in M under K satisfies the temporal geometric predicates of P.

Given a model $M \in \mathcal{M}$ and a scheduler $K \in \mathcal{S}$, we denote by M^K, the set of problems solvable by robots in M under adversarial scheduler K. Let $M_1, M_2 \in \mathcal{M}$ and $K_1, K_2 \in \mathcal{S}$.

- We say that model M_1 under scheduler K_1 is *computationally not less powerful than* model M_2 under K_2, denoted by $M_1^{K_1} \geq M_2^{K_2}$, if $M_1(K_1) \supseteq M_2(K_2)$.
- We say that M_1 under K_1 is *computationally more powerful than* M_2 under K_2, denoted by $M_1^{K_1} > M_2^{K_2}$, if $M_1^{K_1} \geq M_2^{K_2}$ and $(M_1(K_1) \setminus M_2(K_2)) \neq \emptyset$.
- We say that M_1 under K_1 and M_2 under K_2, are *computationally equivalent* , denoted by $M_1^{K_1} \equiv M_2^{K_2}$, if $M_1^{K_1} \geq M_2^{K_2}$ and $M_2^{K_2} \geq M_1^{K_1}$.
- Finally, we say that K_1 K_2, are *computationally orthogonal* (or *incomparable*), denoted by $M_1^{K_1} \bot M_2^{K_2}$, if $(M_1(K_1) \setminus M_2(K_2)) \neq \emptyset$ and $(M_2(K_2) \setminus M_1(K_1)) \neq \emptyset$.

Trivially,

▶ **Lemma 1.** *For any $M \in \mathcal{M}$ and any $K \in \mathcal{S}$:*
1. $M^F \geq M^S \geq M^{A_{LC}} \geq M^A$
2. $M^F \geq M^S \geq M^{A_{CM}} \geq M^{A_M} \geq M^A$
3. $\mathcal{LUMI}^K \geq \mathcal{FSTA}^K \geq \mathcal{OBLOT}^K$
4. $\mathcal{LUMI}^K \geq \mathcal{FCOM}^K \geq \mathcal{OBLOT}^K$

Let us also recall the following equivalence established in [11]:

▶ **Lemma 2** ([11])**.** $\mathcal{LUMI}^A \equiv \mathcal{LUMI}^S$

that is, in the $\mathcal{LUMI}$ model, there is no computational difference between Asynch and Ssynch.

Observe that, in all models, any restriction of the adversarial power of the asynchronous scheduler does not decrease (and possibly increases) the computational capabilities of the robots in that model. In other words, if A_α is a restricted scheduler of A_β, then $M^{A_\beta} \leq M^{A_\alpha}$ for any robot model $M \in \mathcal{M}$.

Note that the difference between A_{CM} and A_M is that there exists just one type of configuration that can be observed in A_M but cannot be observed in A_{CM}: the one before moving but after computing. As for $X \in \{\mathcal{FSTA}, \mathcal{OBLOT}\}$, since robots cannot observe the colors of the other robots, we have $X^{A_{CM}} \equiv X^{A_M}$ and $X^{A_{LC}} \equiv X^A$.

[5] A terminal configuration is one in which, once reached, the robots no longer move.

3 The $\mathcal{OBLOT}$ Computational Landscape

3.1 Separating SSYNCH from ASYNCH

In this section we prove that, under SSYNCH, the robots in $\mathcal{OBLOT}$ are strictly more powerful than under $\mathcal{A}_M$, thus separating SSYNCH from ASYNCH in $\mathcal{OBLOT}$.

To do so, we consider the classical `Collisionless Line Convergence` (`CLCv`) problem, where two robots, r and q, must converge to a common location, moving on the line connecting them, without ever crossing each other; i.e., `CLCv` is defined by the predicate

$$CLC \equiv \Big[\{\exists \ell \in \mathbb{R}^2, \forall \epsilon \geq 0, \exists T \geq 0, \forall t \geq T : |r(t) - \ell| + |q(t) - \ell| \leq \epsilon\},$$
$$\textbf{and } \{\forall t \geq 0 : r(t), q(t) \in \overline{r(0)q(0)}\},$$
$$\textbf{and } \{\forall t \geq 0 : dis(r(0), r(t)) \leq dis(r(0), q(t)), dis(q(0), q(t)) \leq dis(q(0), r(t))\}\Big]$$

and we focus on the monotone version of this problem defined below.

▶ **Definition 3** (MONOTONE LINE CONVERGENCE (`MLCv`)). *The two robots, r and q must solve the* `Collisionless Line Convergence` *problem without ever increasing the distance between them.*

In other words, an algorithm solves `MLCv` iff it satisfies the following predicate:

$$MLC \equiv [CLC \textbf{ and } \{\forall t' \geq t, |r(t') - q(t')| \leq |r(t) - q(t)|\}]$$

First observe that `MLCv` can be solved in $\mathcal{OBLOT}^S$.

▶ **Lemma 4.** `MLCv`$\in \mathcal{OBLOT}^S$. *This holds even under non-rigid movement and in absence of chirality.*

Proof. It is rather immediate to see that the simple protocol using the strategy "move to half distance" satisfies the MLC predicate and thus solves the problem. ◀

On the other hand, `MLCv` is not solvable in $\mathcal{OBLOT}^{A_M}$.

▶ **Lemma 5.** `MLCv`$\notin \mathcal{OBLOT}^{A_M}$ *even under fixed disorientation and agreement on the unit of distance.*

Proof. By contradiction, assume that there exists an algorithm $\mathcal{A}$ that solves `MLCv` in $\mathcal{OBLOT}^{A_M}$. Let the two robots, r and q, have the same unit of distance, initially each see the other on the positive direction of the X axis and their local coordinate system not change during the execution of $\mathcal{A}$. Three observations are in order.

(1) First observe that, by the predicates defining `MLCv`, if a robot moves, it must move towards the other, and in this particular setting, it must stay on its X axis.

(2) Next observe that, every time a robot is activated and executes $\mathcal{A}$, it must move. In fact, if, on the contrary, $\mathcal{A}$ prescribes that a robot activated at some distance d from the other must not move, then, in a fully synchronous execution of $\mathcal{A}$ where both robots are initially at distance d, neither of them will ever move and, thus, will never converge.

(3) Finally observe that, when robot r moves towards q on the X axis after seeing it at distance d, the length $f(d)$ of the computed move is the same as that q would compute if seeing r at distance d.

Consider now the following execution $\mathcal{E}$ under A_M: Initially both robots are simultaneously activated, and are at distance d from each other. Robot r completes its computation and executes the move instantaneously (recall, they are operating under A_M), and continues to be activated and to execute $\mathcal{A}$ while robot q is still in its initial computation.

Each move by r clearly reduces the distance between the two robots. More precisely, by observation (3), after $k \geq 1$ moves, the distance will be reduced from d to d_k where $d_0 = d$ and $d_{k>0} = d_{k-1} - f(d_{k-1}) = d - \sum_{0 \leq i < k} f(d_i)$.

▷ Claim. After a finite number of moves of r, the distance between the two robots becomes smaller that $f(d)$.

Proof. By contradiction, let r never get closer than $f(d)$ to q; that is for every $k > 0$, $d_k > f(d)$.

Consider then the execution $\hat{\mathcal{E}}$ of $\mathcal{A}$ under the RoundRobin synchronous scheduler: the robots, initially at distance d, are activated one per round, at alternate rounds. Observe that, since $\mathcal{A}$ is assumed to be correct under A_M, it must be correct also under RoundRobin. This means that, starting from the initial distance d, for any fixed distance $d' > 0$, the two robots become closer than d'. Let $m(d')$ denote the number of rounds for this to occur; then, the distance between them becomes smaller than $f(d)$ after $m(f(d))$ rounds. Further observe that, after round i, the distance d_i between them is reduced by $f(d_i)$. Summarizing, $d_{m(f(d))} = d - \sum_{0 \leq i < m(f(d))} f(d_i) < f(d)$, contradicting that $d_k > f(d)$ for every $k > 0$. ◁

Consider now the execution $\mathcal{E}$ at the time the distance becomes smaller that $f(d)$; let robot q complete its computation at that time and perform its move, of length $f(d)$, towards r. This move then creates a collision, contradicting the correctness of A. ◀

From Lemmas 4 and 5, and since $\mathcal{OBLOT}^{A_M} \geq \mathcal{OBLOT}^{A}$ by definition, the main result now follows:

▶ **Theorem 6.** $\mathcal{OBLOT}^{S} > \mathcal{OBLOT}^{A}$

In other words, under the synchronous scheduler Ssynch, $\mathcal{OBLOT}$ robots are strictly more powerful than when under the asynchronous scheduler Asynch. This results provides a definite positive answer to the long-open question of whether there exists a computational difference between synchrony and asynchrony in $\mathcal{OBLOT}$.

3.2 Refining the $\mathcal{OBLOT}$ Landscape

We can refine the $\mathcal{OBLOT}$ landscape as follows; By definition, $\mathcal{OBLOT}^{A_M} \geq \mathcal{OBLOT}^{A}$. Consider now the following problem for $n = 4$ robots.

▶ **Definition 7** (TRAPEZOID FORMATION (TF)). *Consider a set of four robots, $R = \{a, b, c, d\}$ whose initial configuration forms a convex quadrilateral $Q = (ABCD) = (a(0)b(0)c(0)d(0))$ with one side, say $\overline{CD}$, longer than all others. The task is to transform Q into a trapezoid T, subject to the following conditions:*

(1) *If Q is a trapezoid, the configuration must stay unchanged (Figure 2(1)); i.e.,*

$$TF1 \equiv [\ Trapezoid(ABCD) \Rightarrow \{\forall t > 0, r \in \{a, b, c, d\} : r(t) = r(0)\}\ ]$$

(2) *Otherwise, without loss of generality, let A be farther than B from CD. Let $Y(A)$ (resp., $Y(B)$) denote the perpendicular lines from A (resp., B) to CD meeting CD in A' (resp. B'), and let α be the smallest angle between $\angle BAA'$ and $\angle ABB'$.*

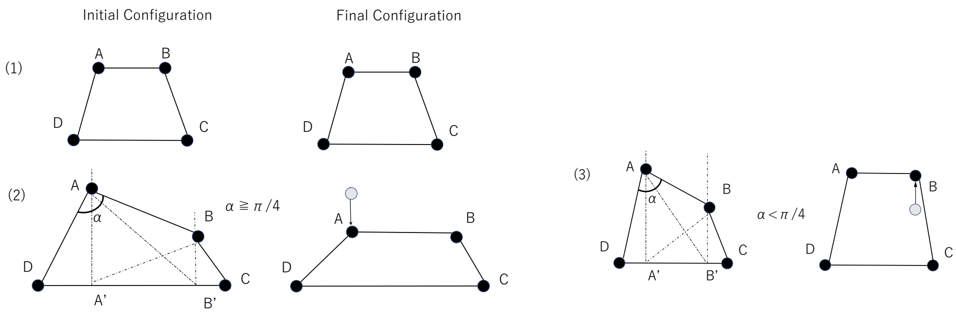

Figure 2 TRAPEZOID FORMATION (TF).

(2.1) *If $\alpha \geq \pi/4$ then the robots must form the trapezoid shown in Figure 2(2), where the location of a is a translation of its initial one on the line $Y(A)$, and that of all other robots is unchanged; specifically,*

$$TF2.1 \equiv [\ (\alpha \geq \pi/4)\ \Rightarrow \{\forall t \geq 0, r \in \{b,c,d\} : r(t) = r(0), a(t) \in Y(A)\}\ \textbf{and}$$
$$\{\exists t > 0 : \forall t' \geq t,\ \{\overline{a(t')b(t')}\ ||\overline{CD}\}\ \textbf{and}\ \{a(t') = a(t)\}\ \}\ ]$$

(2.2) *If instead $\alpha < \pi/4$ then the robots must form the trapezoid shown in Fig. 2(3), where the location of all robots but b is unchanged, and that of b is a translation of its initial one on the line $Y(B)$; specifically,*

$$TF2.2 \equiv [\ (\alpha < \pi/4)\ \Rightarrow \{\forall t \geq 0, r \in \{a,c,d\} : r(t) = r(0), b(t) \in Y(B)\}\ \textbf{and}$$
$$\{\exists t > 0 : \forall t' \geq t,\ \{\overline{a(t')b(t')}\ ||\overline{CD}\}\ \textbf{and}\ \{b(t') = b(t)\}\ \}\ ]$$

Observe that TF can be solved in $\mathcal{OBLOT}^{A_M}$.

▶ **Lemma 8.** *TF $\in \mathcal{OBLOT}^{A_M}$, even in absence of chirality.*

Proof. It is immediate to see that the following simple set of rules solves TF in $\mathcal{OBLOT}^{A_M}$.

Rule 1: If the observed configuration is as shown in Figure 2 (1), the configuration is already a trapezoid, and no robot performs any move ($TF1$).

Rule 2: Let the configuration be as shown in Figure 2 (2). Whenever observed by b, c, d, none of them moves; when observed by a, a moves to the desired point eventually creating a terminal configuration subject to Rule 1. Since the scheduler is *Move*-atomic Asynch the other robots do not observe a during this move, but only after the move is completed.

Rule 3: Analogously, let the configuration be as shown in Figure 2 (3). Whenever observed by a, c, d, none of them moves; when observed by b, b moves to the desired point eventually creating a trapezoid and reaching a terminal configuration, unseen by all other robots during this movement. ◀

However, TF cannot be solved in $\mathcal{OBLOT}^{A}$.

▶ **Lemma 9.** *TF $\notin \mathcal{OBLOT}^{A}$, even with fixed disorientation.*

Proof. By contradiction, let $\mathcal{A}$ be an algorithm that always allows the four $\mathcal{OBLOT}$ robots to solve TF under the asynchronous scheduler. Consider the initial configuration where a is further than b from $\overline{CD}$, and $\alpha = \pi/4$. In this configuration, a is required to move (along $Y(A)$) while no other robot is allowed to move. Observe that, as soon as a moves, it creates a configuration where a is still further than b from $\overline{CD}$, but $\alpha' = \min\{\angle b(t)a(t)A', \angle a(t)b(t)B'\} < \pi/4$.

Consider now the execution of $\mathcal{A}$ in which a is activated first, and then b is activated while a is moving; in this execution, the configuration seen by b requires it to to move, violating $TF2.1$ and contradicting the assumed correctness of algorithm $\mathcal{A}$. ◀

Thus, by Lemmas 8 and 9, we have

▶ **Theorem 10.** $\mathcal{OBLOT}^{A_M} > \mathcal{OBLOT}^{A}$

▶ **Theorem 11.** $\mathcal{OBLOT}^{S} > \mathcal{OBLOT}^{A_M} > \mathcal{OBLOT}^{A_{LC}} \equiv \mathcal{OBLOT}^{A}$

Proof. (1) The equivalence $\mathcal{OBLOT}^{A_{LC}} \equiv \mathcal{OBLOT}^{A}$ holds because, by definition, $\mathcal{OBLOT}$ robots cannot distinguish between A_{LC} and A; then, by Theorem 10, $\mathcal{OBLOT}^{A_M} > \mathcal{OBLOT}^{A_{LC}}$. (2) It follows from Lemmas 4 and 5. (3) It follows from (1) and Theorem 6. ◀

4 The $\mathcal{FCOM}$ Computational Landscape

4.1 Separating Ssynch from Asynch in $\mathcal{FCOM}$

We have seen (Theorem 6) that, to overcome the limitations imposed by asynchrony, the robots must have some additional power with respect to those held in $\mathcal{OBLOT}$.

In this section, we show that the communication capabilities of $\mathcal{FCOM}$ are not sufficient. In fact, we prove that, under Ssynch, the robots in $\mathcal{FCOM}$ are strictly more powerful than under $\mathcal{A}_M$, thus separating Ssynch from Asynch in $\mathcal{FCOM}$. To do so, we use the problem MLCv again.

Observe that `MLCv` can be solved even in $\mathcal{OBLOT}^{S}$ (Lemma 4), and thus in $\mathcal{FCOM}^{S}$.

▶ **Lemma 12.** `MLCv` $\in \mathcal{FCOM}^{S}$; *this holds even under variable disorientation, non-rigid movement and in absence of chirality.*

On the other hand, `MLCv` is not solvable in $\mathcal{FCOM}^{A_M}$.

▶ **Lemma 13.** `MLCv` $\notin \mathcal{FCOM}^{A_M}$.

Proof. Let us consider two robots, r and q, and show that the adversary can activate them in a way that exploits variable disorientation to cause them to violate the condition of `MLCv`.

We consider the execution in which the adversary always forces the robots to perceive the distance between r and q as 1, which is equivalent to the current unit distance of X. We define a function $f(c, d)$ as the length of the move taken by a robot when it observes color c of the other robot and the true distance between the two robots is d in the last Look phase. Since the distance always appears as 1 to the robots, the value $F(c) = f(c, d)/d$ is independent of d. We denote the initial color of the robots as c_0 and assume that $F(c_0) > 0$, which does not affect generality as the adversary can activate r and q multiple times until both robots have a color c such that $F(c) > 0$. Without loss of generality, we also assume that $F(c_0) \leq 1/2$. If $F(c_0) > 1/2$, it follows that r and q pass each other when the adversary activates both robots at time step 0, violating the condition of `MLCv`. Therefore, we assume $0 < F(c_0) < 1/2$ without loss of generality.

Starting from time step 0, the adversary refrains from activating r and instead activates only q to move $\lfloor \log_{1-F(c_0)} F(C_0) \rfloor + 1$ times. Since q always perceives c_0 as the color of r during this period, the distance between r and q decreases by a factor of $(1 - F(C_0))$ with each move of q. As a result, the distance between r and q becomes smaller than $F(C_0)d_0 = f(c_0, d_0)$ after this period, where d_0 is the initial distance between r and q. The adversary then activates r to perform its Move phase. r moves a distance of $f(c_0, d_0)$ and overtakes q, thereby violating the condition. ◀

From Lemmas 12 and 13, and since $\mathcal{FCOM}^{A_M} \geq \mathcal{FCOM}^{A}$ by definition, the main result now follows:

▶ **Theorem 14.** $\mathcal{FCOM}^{S} > \mathcal{FCOM}^{A}$

4.2 Refining the $\mathcal{FCOM}$ Landscape

In this Section we complete the characterization of the asynchronous landscape of $\mathcal{FCOM}$ proving $\mathcal{FCOM}^{A} \equiv \mathcal{FCOM}^{A_M} \equiv \mathcal{FCOM}^{A_{CM}} < \mathcal{FCOM}^{A_{LC}} \equiv \mathcal{FCOM}^{S}$

We first show that every problem solvable by a set of $\mathcal{FCOM}$ robots under A_{CM} can also be solved under Asynch. We do so constructively: we present a *simulation* algorithm for $\mathcal{FCOM}$ robots that allows them to correctly execute in Asynch any protocol $\mathcal{A}$ given in input (i.e., all its executions under Asynch are equivalent to some executions under A_{CM}). The simulation algorithm (called *SIM*) makes each $\mathcal{FCOM}$ robot execute $\mathcal{A}$ infinitely often, never violating the conditions of scheduler CM-**atomic**-Asynch. To achieve this, *SIM* needs an activated robot to be able to retrieve some information about its past (e.g., whether or not it has "recently" executed $\mathcal{A}$). Such information can obviously be encoded and persistently stored by the robot in the color of its own light; but, since an $\mathcal{FCOM}$ robot cannot see the color of its light, the robot cannot access the stored information. However, this information can be seen by the other robots, and hence can be communicated by some of them (via the color of their lights) to the needing robot. This can be done efficiently as follows. Exploiting chirality, the robots can agree at any time on a circular ordering of the nodes where robots are located, so that for any such a location x both its predecessor $\mathtt{pred}(x)$ and its successor $\mathtt{suc}(x)$ in the ordering are uniquely identified, with $\mathtt{pred}(\mathtt{suc}(x)) = x$; all robots located at x then become responsible for communicating the needed information to the robots located at $\mathtt{suc}(x)$[6].

Let $\mathcal{A}$ be an algorithm for $\mathcal{FCOM}$ robots in CM-**atomic**-Asynch, and let $\mathcal{A}$ use a light of ℓ colors: $C = \{c_0, c_1, \ldots c_{\ell-1}\}$. It is assumed that, in any initial configuration $\mathcal{C}$, the number of distinct locations[7] is $m \geq 2$.

As described later, the simulation algorithm is composed of four phases. To execute the simulation protocol, a robot r uses four externally visible persistent lights:

1. $r.light \in C$, indicating its own light used in the execution of $\mathcal{A}$; initially, $r.light = c_0$;
2. $r.phase \in \{1, 2, 3, m\}$, indicating the current phase of the simulation algorithm; initially $r.phase = 1$;
3. $r.state \in \{W, M, F\}$, indicating the state of r in its execution of the simulation; initially, $r.state = W$;
4. $r.\mathtt{suc}.state \in 2^{\{W,M,F\}}$, indicating the set of states at $\mathtt{suc}$(x), where x is the current location of r; initially, r.$\mathtt{suc}$.state=$\{W\}$.

Summarizing, each robot r has $Light[r] = \langle r.light, r.phase, r.state, r.\mathtt{suc}.state\rangle$. For a location x and $l \in \{light, phase, state, \mathtt{suc}.state\}$, let $x.l = \cup_{\mathrm{r\ at\ x}} r.l$ denote the set of the lights $r.l$ of the robots at location x.

Informally, the simulation algorithm is composed of three main phases which are continuously repeated and a fourth one which is occasionally performed. Each execution of the three main phases corresponds to a single execution of $\mathcal{A}$, each satisfying the CM-atomic

6 Although we use chirality to determine the cyclic order, this assumption can be circumvented by slightly increasing the number of light colors and deciding the color of the corresponding robot using local 'suc' and 'pred' [14].

7 In $\mathcal{FCOM}$, by definition, if all robots of the same color are located on the same position, they would not be able to see anything including themselves and they could not perform any task.

condition, by some robots. The three main phases are repeated until every robot has executed $\mathcal{A}$ at least once, ensuring fairness. Appropriate flags are set up to detect this occurrence; a "mega-cycle" is said to be completed, and after the execution of the fourth phase (a reset), a new mega-cycle is started (continuing the simulation of the execution of $\mathcal{A}$ through the continuing execution of the three phases). In other words, in each mega-cycle all robots are activated and execute[8] $\mathcal{A}$ under the CM-**atomic** condition.

Let us describe the protocol in more details. After the *Look* operation, an activated robot r at x recognize its own $r.state$ by using the predecessor's $\texttt{suc}.state$ and the set $r.state.here$ of states seen by r at its own location x. It understands to be in Phase 1 by detecting $\rho.phase = 1$ for any other robot ρ.

In Phase 1, if the activated robot r has not executed yet $\mathcal{A}$ in the current mega-cycle (its state is W) and does not observe any robot with $state = M$, r changes its state flag from W to M and executes $\mathcal{A}$; otherwise, it changes $r.phase$ to 2. Note that, in Phase 1 only a robot with state flag M might be moving; hence since activated robots do not execute $\mathcal{A}$ if they see any robot in phase 1 with state flag M, the simulated algorithm is executed under CM-**atomic**-Asynch. Once all these executions of $\mathcal{A}$ are completed, within finite time every robot r has $r.phase = 2$, and the second phase starts.

In Phase 2, there are no robots executing algorithm $\mathcal{A}$, no robot is moving, and the locations of the robots remains unchanged. An activated robot r at x only updates $r.\texttt{suc}.state$ by observing $\texttt{suc}(x)$ and change its *phase* flag from 2 to 3. When every robot r has $r.phase$ to 3, the third phase starts.

In Phase 3, if robot r executed algorithm $\mathcal{A}$ in Phase 1 ($r.state = M$), then it changes its state flag from M to F (to ensure that the scheduling of robots performing the simulated algorithms is fair). After all robots with M change their state flags to F, every robot copies its neighboring states' flags ($\texttt{suc}(x).state$) setting Phase to 1.

At the beginning of Phase 1, the end of the current mega-cycle is checked. If the mega-cycle is finished (i.e., all robots have their state flags set to F), the robots enter the special Phase m, in which each robot r resets the flag $r.state = W$ and $r.\texttt{suc}.state = \{W\}$. Once completed, the robots return to Phase 1 and begin a new mega-cycle. Observe that, during the transition from Phase 1 to Phase m and from Phase m back to Phase 1, there are configurations containing both phase flags m and 1; it is however not difficult for the robots to distinguish the particular transition being observed: if $\forall r(r.state = F)$is true, it is the former, otherwise $\forall r(r.state = W)$ is true it is the latter.

The correctness of the simulation (see [19]) implies the following:

▶ **Theorem 15.** *$\mathcal{FCOM}^{A_{CM}} \equiv \mathcal{FCOM}^{A}$. This holds even in absence of chirality.*

Since $A_{CM} \le A_M \le A$, in turn this theorem implies the following corollary.

▶ **Corollary 16.** *$\mathcal{FCOM}^{A_M} \equiv \mathcal{FCOM}^{A}$. This holds even in absence of chirality.*

We can use the same simulation algorithm to show the equivalence between A_{LC} and Ssynch in $\mathcal{FCOM}$. Since $\mathcal{FCOM}^{A_{LC}} \le \mathcal{FCOM}^{S}$ by definition, to prove $\mathcal{FCOM}^{A_{LC}} \equiv \mathcal{FCOM}^{A_M}$, we need to show that every problem solvable by a set of $\mathcal{FCOM}$ robots under Ssynch can also be solved under A_{LC}.

The *simulation* algorithm for $\mathcal{FCOM}$ robots that allows them to correctly execute under A_{LC} any protocol designed to work under Ssynch, is actually precisely the simulation algorithm *SIM* described in the previous subsection, that executes under Asynch any

[8] In each phase of mega-cycles, at most one robot may execute the simulated algorithm more than once.

Algorithm 1 `SIM(A)`: predicates and subroutines for robot r at location x.

Assumptions: Let $x_0, x_1, \ldots, x_{m-1}$ be the circular arrangement on the configuration C ($m \geq 2$), and let define $\texttt{suc}(x_i) = x_{i+1 \mod m}$ and $\texttt{pred}(x_i) = x_{i-1 \mod m}$.

State Look
Observe, in particular, $\texttt{pred}(x).state(\texttt{suc}.state)$, $\texttt{suc}(x).state(\texttt{suc}.state)$, and $\rho.phase(\rho \neq r)$; as well as $r.state.here$ (the set of states seen by r at its own location x.
Note that, for this, r cannot see its own color).

predicate Is-all-phases(p:phase)
$\forall \rho (\neq r)(\rho.phase = p)$

predicate Is-phases-mixed(p, q: phase)
$\forall \rho (\neq r)[(\rho.phase = p)$ **or** $(\rho.phase = q)]$
and [**not** *Is-all-phases(p)*] **and** [**not** *Is-all-phases(q)*]

predicate Is-exist-M
$\exists \rho (\neq r)[(\rho.state = M)$ **or** $(M \in \rho.suc.state)]$

predicate Is-all(s: state)
$\forall \rho(\rho.state = s)$ **and** $own.state = \{s\}$

function $r.own.state$: set of states
 $own.state \leftarrow \texttt{pred}(x).\texttt{suc}.state - x.state.here$,
 where $x.state.here$ corresponds to the set of states seen by r at its own location x

subroutine Copy-States-of-Neighbors
 r.$\texttt{suc}$.state $\leftarrow \texttt{suc}(x).state$

subroutine Reset-state-and-neighbor-state
 $r.state \leftarrow W$
 $r.\texttt{suc}.state \leftarrow \{W\}$

---

algorithm $\mathcal{A}$ designed to work under A_{CM}. To understand why this is the case, observe that, as we have shown, *any* asynchronous execution of SIM with algorithm $\mathcal{A}$ in input produces a specific execution of $\mathcal{A}$ under A_{CM}. If the executions of SIM were not arbitrary (i.e., under Asynch) but under a restricted asynchronous scheduler (say A_α), then each such execution would clearly produce a specific execution of $\mathcal{A}$ under the asynchronous scheduler which satisfies both CM and α.

Thus, the execution of SIM under A_{LC} with $\mathcal{A}$ in input, will produce an execution of $\mathcal{A}$ that satisfies both LC and CM; that is, an execution under LCM-**atomic**-Asynch = Ssynch.

▶ **Theorem 17.** $\mathcal{FCOM}^S \equiv \mathcal{FCOM}^{A_{LC}}$.

Finally, Theorems 14–17 imply the following separation:

▶ **Theorem 18.** $\mathcal{FCOM}^{A_{LC}} > \mathcal{FCOM}^{A_{CM}}$.

Algorithm 2 `SIM(A)` - for robot r at location x.

State Compute

r.des $\leftarrow$ r.pos
if *Is-all-phases(1)* **then**
  $Copy\text{-}state\text{-}of\text{-}Neighbors$
  $r.phase \leftarrow 1$
  if *Is-all(F)* **then** $r.phase \leftarrow m$
  else if $(\exists \rho (\neq r)[(\rho.state = M))$ **then** $r.phase \leftarrow 2$
  else if $(r.own.state = \{W\})$ **then**
    Execute the `Compute` of $\mathcal{A}$ // determining my color $r.light$ and destination $r.des$ //
    $r.state \leftarrow M$
else if *Is-all-phases(2)* **then**
  $r.phase \leftarrow 3$
  $Copy\text{-}state\text{-}of\text{-}Neighbors$
else if *Is-all-phases(3)* **then**
  $Copy\text{-}state\text{-}of\text{-}Neighbors$
  $r.phase \leftarrow 3$
  if *Is-exist-M* **then**
    if $r.own.state = \{M\}$ **then**
      $r.state \leftarrow F$
    $Copy\text{-}state\text{-}of\text{-}Neighbors$
  else// *no-M*//
    $r.phase \leftarrow 1$
    $Copy\text{-}state\text{-}of\text{-}Neighbors$
else if *Is-phase-mixed(1,2)* **then**
    $r.phase \leftarrow 2$
else if *Is-phase-mixed(2,3)* **then**
    $r.phase \leftarrow 3$
    $Copy\text{-}state\text{-}of\text{-}Neighbors$
else if *Is-phase-mixed(1,3)* **then**
    $r.phase \leftarrow 1$
    $Copy\text{-}state\text{-}of\text{-}Neighbors$
else if *Is-all-phases(m)* **then** //Reset state//
  Reset-state-and-neighbor-state
  if $\exists \rho \neq r(\rho.state = F)$ **then**
    r.phase $\leftarrow m$
  else //There does not exist F//
    r.phase $\leftarrow 1$
else if *Is-phase-mixed(1,m)* **and** *Is-all(F)* **then** $r.phase \leftarrow m$
else if *Is-phase-mixed(1,m)* **and** *Is-all(W)* **then** $r.phase \leftarrow 1$

State Move

Move to $r.des$;

5 The $\mathcal{FSTA}$ Computational Landscape

5.1 Separating SSYNCH from ASYNCH in $\mathcal{FSTA}$

In this section, we consider the $\mathcal{FSTA}$ model; in this model, the only difference with $\mathcal{OBLOT}$ is that the robots are endowed with a bounded amount of memory whose content persists from a cycle to the next. We investigate whether, with this additional capability, the robots are able to overcome the limitations imposed by asynchrony, The answer is unfortunately negative: we prove that, also in this model, the otherwise enhanced robots are strictly more powerful under the synchronous scheduler SSYNCH than under the asynchronous one ASYNCH.

To do so, we consider the problem MLCv again.

Observe that MLCv can be solved even in $\mathcal{OBLOT}^S$ (Lemma 4), and thus in $\mathcal{FSTA}^S$.

▶ **Lemma 19.** MLCv $\in \mathcal{FSTA}^S$; *this holds even under variable disorientation, non-rigid movement and in absence of chirality.*

On the other hand, MLCv cannot be solved in $\mathcal{FSTA}^{A_M}$.

▶ **Lemma 20.** MLCv $\notin \mathcal{FSTA}^{A_M}$

Proof. Let r and q be the two robots that we consider. In what follows, we will show that for any algorithm, the adversary can activate r and q and exploit variable disorientation so that they violate the condition of MLCv.

Because of variable disorientation, whenever a robot $X \in \{r, q\}$ performs a Look operation, the adversary can (and will) force the *observed* distance between r and q in the resulting snapshot to be always 1 (i.e., equals the current unit distance of X). Let $f(c, d)$ be the length of the computed move when a robot has color c and the *real* distance between the two robots is d in the last Look phase. Note that $F(c) = f(c, d)/d$ does not depend on d because the distance always looks one to the robots.

Since the distance always looks the same to the robots, unless the two robots meet, the transition sequence of the internal colors set by a robot is fixed. In particular, since the number of colors is a fixed constant, after a finite transient, say $(c_0, c_1, \ldots, c_k)$, the sequence becomes periodic, say $(c_s, c_{s+1}, ..., c_k)^*$.

Then, the adversary can activate the robots in the following way so that, either during the transient they violate the condition of MLCv, or both of them end the transient without meeting each other and have color c_s:

1. $i \leftarrow 0$.
2. If $F(c_i) > 1/2$, the adversary activates both r and q, by which they pass each other, clearly violating the condition of MLCv. If $F(c_i) \leq 1/2$, the adversary first activates r, and then activates q, by which r and q never meets (i.e., never reach the same location).
3. $i \leftarrow i + 1$ and go back to 2.

If the robots did not violate the condition of MLCv during their transient, they are both at the beginning of their periodic sequence with color c_s in distinct positions. If $F(c_i) = 0$ holds for all $i = s, s+1, \ldots, k$, no robot moves, thus MLCv is never solved. So, without loss of generality, we assume $c_s = c_0$ and $F(c_0) > 0$. Then, the following strategy of the adversary leads to the violation of the condition of MLCv, where d_0 is the distance between r and q at time 0.

1. Let r and q perform *Look* and *Compute* phase, by which both r and q compute to move by distance $f(c_0, d_0)$.
2. While r is still waiting to be activated to move, activate only q repeatedly until q overtakes r or the distance between r and q becomes less than $f(c_0, d_0)$. The former case occurs if $F(c_i) > 1$ for some i. This obviously violates the condition of MLCv. Otherwise, the latter case must eventually occur because the distance between r and q becomes constant times smaller each time q changes its color k times. Then, the adversary finally activate r to perform its Move phase. Then, r moves a distance $f(c_0, d_0)$ and overtakes q, violating the condition.

Thus, for any algorithm, the two robots must violate the condition of MLCv. ◀

Thus, by Lemmas 19 and 20, a separation between SSYNCH and ASYNCH in $\mathcal{FSTA}$ is shown.

▶ **Theorem 21.** $\mathcal{FSTA}^S > \mathcal{FSTA}^A$

5.2 Refining the $\mathcal{FSTA}$ landscape

We can refine the $\mathcal{FSTA}$ landscape as follows; Consider again the TF problem defined and analyzed in Section 3.2. By Lemma 8, TF can be solved in $\mathcal{OBLOT}^{A_M}$, and thus in $\mathcal{FSTA}^{A_M}$.

On the other hand, TF is not solvable in $\mathcal{FSTA}^{A_{LC}}$.

▶ **Lemma 22.** *TF* $\notin \mathcal{FSTA}^{A_{LC}}$*, even with fixed disorientation.*

Proof. By contradiction, let $\mathcal{A}$ be an algorithm that always allows the two $\mathcal{FSTA}$ robots to solve TF and form a trapezoid reaching a terminal state in finite time under the A_{LC} scheduler. Consider the initial configuration where a is further than b from $\overline{CD}$, and $\alpha = \pi/4$. Starting from this configuration, a is required to move within finite time along $Y(A)$; on the other hand, no other robot is allowed to move. Consider now the execution of $\mathcal{A}$ in which only a is activated, and starts moving at time t; observe that, as soon as a moves, it creates a configuration where a is still further than b from $\overline{CD}$, but $\alpha' = \min\{\angle b(t)a(t)A', \angle a(t)b(t)B'\} < \pi/4$.

Activate now b at time $t' > t$ while a is still moving. Should this have been an initial configuration, within a constant number of activations (bounded by the number of internal states), b would move, say at time t". In the current execution, slow down the movement of a so that it is still moving at time t''. Since in $\mathcal{FSTA}$ b cannot access the internal state of a, nor remember previously observed angles and distances, it cannot detect that the observed configurations are not initial configurations; hence it will move at time t", violating $TF2$ and contradicting the assumed correctness of $\mathcal{A}$. ◀

Summarizing: by definition, $\mathcal{FSTA}^{A_M} \geq \mathcal{FSTA}^A$; by Lemma 8, it follows that TF is solvable in $\mathcal{FSTA}^{A_M}$; and, by Lemma 22, it follows that TF is not solvable in $\mathcal{FSTA}^A$. In other words:

▶ **Theorem 23.** $\mathcal{FSTA}^{A_M} > \mathcal{FSTA}^A$

▶ **Theorem 24.**
1. $\mathcal{FSTA}^{A_{LC}} \equiv \mathcal{FSTA}^A$
2. $\mathcal{FSTA}^S > \mathcal{FSTA}^{A_M}$
3. $\mathcal{FSTA}^S > \mathcal{FSTA}^{A_{LC}}$
4. $\mathcal{FSTA}^{A_M} > \mathcal{FSTA}^{A_{LC}}$

Proof. **1.** holds because, by definition, $\mathcal{FSTA}$ robots cannot distinguish between A_{LC} and A. **2.** follows from follows from Lemmas 19 and 20. **3.** follows from **1.** and Theorem 21. **4.** follows from **1.** and Theorem 23. ◀

6 Relationship Between Models Under Asynchronous Schedulers

In the previous sections, we have characterized the asynchronous landscape within each robot model. In this section, we determine the computational relationship between the different models under the asynchronous schedulers A_{LC}, A_M, A_{CM} and ASYNCH.

We do so by first determining the relationship between $\mathcal{FCOM}$ and the other models under the asynchronous schedulers; we then complete the characterization of the landscape by establishing the still remaining relationships, those between $\mathcal{FSTA}$ and $\mathcal{OBLOT}$.

6.1 Relative power of $\mathcal{FCOM}$

In this section, we determine the relationship between $\mathcal{FCOM}$ and the other models under the asynchronous schedulers A_{LC}, A_M, A_{CM} and Asynch.

We first show that $\mathcal{FCOM}^{A_{LC}}$ and $\mathcal{FSTA}^{A_M}$ are orthogonal. To prove this result we use the existence of a problem, `Cyclic Circles` (CYC), shown in [6] to be solvable in $\mathcal{FCOM}^A$ but not in $\mathcal{FSTA}^S$:

▶ **Lemma 25** ([6]).
1. *$CYC \notin \mathcal{FSTA}^S$*
2. *$CYC \in \mathcal{FCOM}^A$, even under non-rigid-movement.*

We then consider the problem `Get Closer but Not too Close on Line` (GCNCL) defined as follows.

▶ **Definition 26** (Get Closer but Not too Close on Line (GCNCL)). *Let a and b be two robots on distinct locations $a(0), b(0)$ where $r(t)$ denotes the position of $r \in \{a, b\}$ at time $t \geq 0$. This problem requires the two robots to get closer, without ever increasing their distance on the line connecting them, and eventually stop at distance at least $|a(0) - b(0)|/2$ from each other.*

In other words, an algorithm solves GCNCL iff it satisfies the following predicate:

$$GCNCL \equiv \left(\forall t \geq 0 : a(t), b(t) \in \overline{a(0)b(0)}\right) \wedge (\forall t, t' : 0 \leq t \leq t' \to d_t \geq d_{t'})$$
$$\wedge \left(\exists t : \frac{d_0}{2} \leq d_t < d_0 \wedge (\forall t' \geq t : a(t) = a(t') \wedge b(t) = b(t'))\right),$$

where d_t is the distance between the two robots at time t, i.e., $d_t = |a(t) - b(t)|$.

▶ **Lemma 27.**
1. *$GCNCL \notin \mathcal{FCOM}^S$.*
2. *$GCNCL \in \mathcal{FSTA}^A$.*

Proof. **1.** The impossibility of $\mathcal{FCOM}^S$ can be obtained as follows. Since we consider $\mathcal{FCOM}$, a robot computes its destination depending on the color of its opponent, not on its own color. We say that a color c is *attractive* if a robot decides to move (i.e., not stay) when the color of the opponent is c. The adversary can prevent the robots from solving GCNCL in the following way. Initially, both robots have the same color. If that color is not attractive, the adversary keeps on simultaneously activating both robots until the color of the robots becomes attractive. During this period, no robot moves by the definition of attractive colors. Note that an attractive color must appear eventually to solve GCNCL. From then on, the adversary keeps on activating only one robot, say a, while never activating b. During this period, b never changes its color, so the color of b is always attractive. Because of variable disorientation, the adversary can guarantee that there is a fixed positive constant $c \leq 1$ such that when a is activated at time t, the resulting distance between a and b (after a moves) is $c \cdot d_t = c \cdot |a(t) - b(t)|$. (The robots immediately violate the specification of GCNCL if $c > 1$ or $c = 0$.) However, this implies that the distance between a and b converges to zero as a moves repeatedly, violating the specification of GCNCL.

2. The problem is easily solvable with $\mathcal{FSTA}$ robot in Asynch. Let the robots have color A initially. The first time a robot is activated, it moves closer by distance $d/4$ to the other and changes its color to B, where d is the observed distance. Whenever a robot is activated, if its color is B, it does not move. Clearly, both robots eventually stop and their final distance is at least $d_0/2$. ◀

The orthogonality of $\mathcal{FCOM}^{A_{LC}}$ and $\mathcal{FSTA}^{A_M}$ (or $\mathcal{FSTA}^{A}$) then follows from Lemmas 25 and 27.

▶ **Theorem 28.**
1. $\mathcal{FCOM}^{A_{LC}} \perp \mathcal{FSTA}^{A_M}$
2. $\mathcal{FCOM}^{A_{LC}} \perp \mathcal{FSTA}^{A}$
3. $\mathcal{FCOM}^{A_{LC}} > \mathcal{OBLOT}^{S}$

Proof. **1.**–**2.** By Lemmas 25 and 27. **3.** is proved by the fact that RDV can be solved by $\mathcal{FCOM}^{S}$ but not by $\mathcal{OBLOT}^{S}$, and by the equivalence of $\mathcal{FCOM}^{S}$ and $\mathcal{FCOM}^{A_{LC}}$. ◀

The following theorem shows the relative power of $\mathcal{FCOM}^{A}$.

▶ **Theorem 29.**
1. $\mathcal{FCOM}^{A} (\equiv \mathcal{FCOM}^{A_M}) \perp \mathcal{FSTA}^{A_M}$
2. $\mathcal{FCOM}^{A} \perp \mathcal{FSTA}^{A}$
3. $\mathcal{FCOM}^{A} \perp \mathcal{OBLOT}^{S}$
4. $\mathcal{FCOM}^{A} > \mathcal{OBLOT}^{A_M}$

Proof. **1.** (resp. **2.**) follows from Theorem 28 **1.** (resp. **2.**) and noting that CYC can be solved in $\mathcal{FCOM}^{A}$. **3.** is proved by Lemmas 4 and 13(MLCv can be solved in $\mathcal{OBLOT}^{S}$ but cannot be solved in $\mathcal{FCOM}^{A_M}$) and the fact that CYC cannot be solved in $\mathcal{FSTA}^{S}$ (and so $\mathcal{OBLOT}^{S}$). **4.** is proved by the equivalence of $\mathcal{FCOM}^{A_M}$ and $\mathcal{FCOM}^{A}$ and using the result of RDV. ◀

The relationship between $\mathcal{FCOM}$ and the other models under the asynchronous schedulers has been determined in the previous section (Theorems 28 and 29). To complete the characterization of the relationship between the computational power of the models under the asynchronous schedulers, we need to determine the relationship between $\mathcal{FSTA}$ and $\mathcal{OBLOT}$.

▶ **Theorem 30.**
1. $\mathcal{FSTA}^{A_M} \perp \mathcal{OBLOT}^{S}$
2. $\mathcal{FSTA}^{A_M} > \mathcal{OBLOT}^{A_M} > \mathcal{OBLOT}^{A}$
3. $\mathcal{FSTA}^{A} \perp \mathcal{OBLOT}^{A_M}$
4. $\mathcal{FSTA}^{A} \perp \mathcal{OBLOT}^{S}$
5. $\mathcal{FSTA}^{A} > \mathcal{OBLOT}^{A}$

Proof. Note that RDV can be solved in $\mathcal{FSTA}^{A}$ (and so $\mathcal{FSTA}^{A_M}$) but cannot be solved in $\mathcal{OBLOT}^{S}$ (and so $\mathcal{OBLOT}^{A_M}$ and $\mathcal{OBLOT}^{A}$). **1.** is proved by the results of RDV, and MLCv, which can be solved in $\mathcal{OBLOT}^{S}$ but cannot be solved in $\mathcal{FSTA}^{A_M}$ (Lemmas 4 and 20). **2.** is proved with the result of RDV and Theorem 10. **3.** (resp. **4.**) are proved with the result of RDV and TF (Lemmas 8, 22 and the equivalence of $\mathcal{FSTA}^{A_{LC}}$ and $\mathcal{FSTA}^{A}$) (resp. MLCv (Lemmas 4, 20 and Theorem 23)). **5.** is proved by the result of RDV. ◀

7 Concluding Remarks

In this paper, we investigated the computational relationship between the power of the four models $\mathcal{OBLOT}$, $\mathcal{FSTA}$, $\mathcal{FCOM}$ and $\mathcal{LUMI}$, under a range of asynchronous schedulers, from SSYNCH to ASYNCH, and provided a complete characterization of such relationships. In this process, we have established a variety of results on the computational powers of the robots in presence or absence of (limited) internal capabilities of memory persistence and/or communication. These results include the proof of computational separation between SSYNCH and ASYNCH in absence of either capability, closing several important open questions.

This investigation has also provided valuable insights into the elusive nature of the relationship between asynchrony and the level of atomicity of the *Look*, *Comp*, and *Move* operations performed in an *LCM* cycle. In fact, in this paper, the study of the asynchronous landscapes has focused on precisely the set of asynchronous schedulers defined by the different possible atomic combinations of those operations as well as the *Move* operation: starting from *LCM*-**atomic**-Asynch, which corresponds to Ssynch, ending with Asynch, and including *LC*-**atomic**-Asynch, *CM*-**atomic**-Asynch, and *M*-**atomic**-Asynch.

These results open several new research directions. In particular, an important direction is the examination of other classes of asynchronous schedulers, to further understand the nature of asynchrony for robots operating in *LCM* cycles, identify the crucial factors that render asynchrony difficult for the robots, and possibly discover new methods to overcome it.

References

1. N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006. doi:10.1137/050645221.
2. H. Ando, Y. Osawa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visivility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.
3. S. Bhagat and K. Mukhopadhyaya. Optimum algorithm for mutual visibility among asynchronous robots with lights. In *Proc. 19th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 341–355, 2017. doi:10.1007/978-3-319-69084-1_24.
4. Z. Bouzid, S. Das, and S. Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *the 33rd Int. Conf. on Distributed Computing Systems*, pages 337–346, 2013. doi:10.1109/ICDCS.2013.27.
5. K. Buchin, P. Flocchini, I. Kostitsyna, T. Peters, N. Santoro, and K. Wada. Autonomous mobile robots: Refining the computational landscape. In *APDCM 2021*, pages 576–585, 2021. doi:10.1109/IPDPSW52791.2021.00091.
6. K. Buchin, P. Flocchini, I. Kostitsyna, T. Peters, N. Santoro, and K. Wada. On the computational power of energy-constrained mobile robots: Algorithms and cross-model analysis. In *Proc. 29th Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 42–61, 2022. doi:10.1007/978-3-031-09993-9_3.
7. D. Canepa and M. Potop-Butucaru. Stabilizing flocking via leader election in robot networks. In *Proc. 10th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 52–66, 2007. doi:10.1007/978-3-540-76627-8_7.
8. S. Cicerone, Di Stefano, and A. Navarra. Gathering of robots on meeting-points. *Distributed Computing*, 31(1):1–50, 2018. doi:10.1007/S00446-017-0293-3.
9. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing*, 41(4):829–879, 2012. doi:10.1137/100796534.
10. R. Cohen and D. Peleg. Convergence properties of the gravitational algorithms in asynchronous robot systems. *SIAM J. on Computing*, 34(6):1516–1528, 2005. doi:10.1137/S009753970444647.
11. S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. Autonomous mobile robots with lights. *Theoretical Computer Science*, 609:171–184, 2016. doi:10.1016/J.TCS.2015.09.018.
12. G.A. Di Luna, P. Flocchini, S.G. Chaudhuri, F. Poloni, N. Santoro, and G. Viglietta. Mutual visibility by luminous robots without collisions. *Information and Computation*, 254(3):392–418, 2017. doi:10.1016/J.IC.2016.09.005.
13. S. Dolev, S. Kamei, Y. Katayama, F. Ooshita, and K. Wada. Brief announcement: Neighborhood mutual remainder and its self-stabilizing implementation of look-compute-move robots. In *33rd International Symposium on Distributed Computing*, pages 43:1–43:3, 2019. doi:10.4230/LIPICS.DISC.2019.43.

14 P. Flocchini, G. Prencipe, and N. Santoro (Eds). *Distributed Computing by Mobile Entities.* Springer, 2019. doi:10.1007/978-3-030-11072-7.

15 P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots.* Morgan & Claypool, 2012. doi:10.2200/S00440ED1V01Y201208DCT010.

16 P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard tasks for weak robots: the role of common knowledge in pattern formation by autonomous mobile robots. In *10th Int. Symp. on Algorithms and Computation (ISAAC)*, pages 93–102, 1999. doi:10.1007/3-540-46632-0_10.

17 P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1–3):147–169, 2005. doi:10.1016/J.TCS.2005.01.001.

18 P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science*, 407:412–447, 2008. doi:10.1016/J.TCS.2008.07.026.

19 P. Flocchini, N. Santoro, Y. Sudo, and K. Wada. On asynchrony, memory, and communication: Separations and landscapes. CoRR abs/2311.03328, arXiv, 2023. arXiv:2311.03328.

20 P. Flocchini, N. Santoro, G. Viglietta, and M. Yamashita. Rendezvous with constant memory. *Theoretical Computer Science*, 621:57–72, 2016. doi:10.1016/J.TCS.2016.01.025.

21 P. Flocchini, N. Santoro, and K. Wada. On memory, communication, and synchronous schedulers when moving and computing. In *Proc. 23rd Int. Conference on Principles of Distributed Systems (OPODIS)*, pages 25:1–25:17, 2019. doi:10.4230/LIPICS.OPODIS.2019.25.

22 N. Fujinaga, Y. Yamauchi, H. Ono, S. Kijima, and M. Yamashita. Pattern formation by oblivious asynchronous mobile robots. *SIAM Journal on Computing*, 44(3):740–785, 2015. doi:10.1137/140958682.

23 V. Gervasi and G. Prencipe. Coordination without communication: The case of the flocking problem. *Discrete Applied Mathematics*, 144(3):324–344, 2004. doi:10.1016/J.DAM.2003.11.010.

24 A. Hériban, X. Défago, and S. Tixeuil. Optimally gathering two robots. In *Proc. 19th Int. Conference on Distributed Computing and Networking (ICDCN)*, pages 1–10, 2018. doi:10.1145/3154273.3154323.

25 T. Izumi, S. Souissi, Y. Katayama, N. Inuzuka, X. Défago, K. Wada, and M. Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal on Computing*, 41(1):26–46, 2012. doi:10.1137/100797916.

26 D. Kirkpatrick, I. Kostitsyna, A. Navarra, G. Prencipe, and N. Santoro. Separating bounded and unbounded asynchrony for autonomous robots: Point convergence with limited visibility. In *40th Symposium on Principles of Distributed Computing (PODC)*. ACM, 2021. doi:10.1145/3465084.3467910.

27 T. Okumura, K. Wada, and X. Défago. Optimal rendezvous $\mathcal{L}$-algorithms for asynchronous mobile robots with external-lights. In *Proc. 22nd Int. Conference on Principles of Distributed Systems (OPODIS)*, pages 24:1–24:16, 2018. doi:10.4230/LIPICS.OPODIS.2018.24.

28 T. Okumura, K. Wada, and Y. Katayama. Brief announcement: Optimal asynchronous rendezvous for mobile robots with lights. In *Proc. 19th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 484–488, 2017. doi:10.1007/978-3-319-69084-1_36.

29 G. Sharma, R. Alsaedi, C. Bush, and S. Mukhopadyay. The complete visibility problem for fat robots with lights. In *Proc. 19th Int. Conference on Distributed Computing and Networking (ICDCN)*, pages 21:1–21:4, 2018. doi:10.1145/3154273.3154319.

30 S. Souissi, T. Izumi, and K. Wada. Oracle-based flocking of mobile robots in crash-recovery model. In *Proc. 11th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 683–697, 2009. doi:10.1007/978-3-642-05118-0_47.

31 I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28:1347–1363, 1999. doi:10.1137/S009753979628292X.

32 S. Terai, K. Wada, and Y. Katayama. Gathering problems for autonomous mobile robots with lights. *Theoretical Computer Science*, 941(4):241–261, 2023. doi:10.1016/J.TCS.2022.11.018.

33 G. Viglietta. Rendezvous of two robots with visible bits. In *10th Int. Symp. on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS)*, pages 291–306, 2013. doi:10.1007/978-3-642-45346-5_21.

34 M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26–28):2433–2453, 2010. doi:10.1016/J.TCS.2010.01.037.

35 Y. Yamauchi, T. Uehara, S. Kijima, and M. Yamashita. Plane formation by synchronous mobile robots in the three-dimensional euclidean space. *J. ACM*, 64:3(16):16:1–16:43, 2017. doi:10.1145/3060272.