Verified Planar Formation Control Algorithms by Composition of Primitives

Leonardo Bobadilla[*]

Florida International University, Miami, Florida 33199, USA.

Taylor T. Johnson[†]

University of Texas at Arlington, Arlington, Texas 76019, USA.

Amy LaViers[‡] and Umer Huzaifa

University of Virginia, Charlottesville, Virginia 22904, USA.

Movement primitives and formal methods have been proposed for many robotic applications. In this paper, we discuss work-in-progress utilizing formal methods for synthesizing high-level specifications written in linear temporal logic (LTL) realized with low-level primitives that ensure these specifications produce physically feasible swarm behaviors. The methodology synthesizes higher-level, choreographed behaviors for virtual kinematic chains of planar mobile robots that are realized with primitives consisting of (a) a one-dimensional distributed flocking algorithm with verified properties and (b) planar homogeneous transformations (rotations and translations). We show how to use the methodology to construct verified distributed algorithms for higher-dimensional (planar) shape formation. The existing one-dimensional algorithm has two main properties: (safety) avoidance of collisions between swarm members, and (progress) eventual flock (platoon) formation, which in one dimension is a roughly equal spacing between adjacent robots. By combining this one-dimensional flocking algorithm with other simple local operations, namely rotations and other distributed consensus (averaging) algorithms, we show how to create planar formations with planar safety and progress properties.

Nomenclature

N	Number of agents (robot) in swarm system
$[\mathsf{N}]$	Set of unique robot identifiers
$x_i[k]$	x-axis Position of robot i at time step k
$y_i[k]$	y-axis Position of robot i at time step k
$\hat{x}_i[k]$	x-axis Position of robot i at time step k in rotated reference frame
$\hat{y}_i[k]$	y-axis Position of robot i at time step k in rotated reference frame
$\theta_i[k]$	Rotation angle for rotated reference frame of robot i at time step k
$A_i^\theta[k]$	Rotation matrix using $\theta_i[k]$ for robot i at time step k
$L(i)$	Identifier of robot left of robot i in rotated reference frame, if any
$R(i)$	Identifier of robot right of robot i in rotated reference frame, if any
$\mathcal{G}$	Set of agent groups that partition $[\mathsf{N}]$
r_s	Safety spacing
r_f	Nominal flocking spacing

[*]Assistant Professor, Computing and Information Sciences, Florida International University; co-first author.
[†]Assistant Professor, Computer Science and Engineering, University of Texas at Arlington; co-first author.
[‡]Assistant Professor, Systems and Information Engineering, University of Virginia; co-first author.

r_{ti}	Tight flocking spacing preset
r_{lo}	Loose flocking spacing preset
θ_{ac}	Acute group angle preset
θ_{ob}	Obtuse group angle preset
$\neg$	Boolean operator, negation (not)
$\vee$	Boolean operator, disjunction (or)
$\wedge$	Boolean operator, conjunction (and)
$\rightarrow$	Boolean operator, implication
$\mathbf{X}$	Temporal operator, next
$\mathcal{U}$	Temporal operator, until
$\mathbf{F}$	Temporal operator, eventually
$\mathbf{G}$	Temporal operator, always
Π	Set of propositions
ϕ	Linear temporal logic formula (or specification)

I. Introduction

Developing robust algorithms with verified properties for distributed robotics systems is challenging, but there are robust methods for forming flocks [25] and planar vee's (wedges) [2, 7], as well as one-dimensional platoons that are safe (verified collision avoidance) with multiple platoons and platoon merging [15]. There are many algorithms and methods for the formation of shapes like flocks [25, 26] and planar vee's (wedges) [2]. The one-dimensional flocking algorithm used in this paper was inspired by [12], developed into a safe and self-stabilizing (fault-tolerant) with multiple groups of robots and group merging in [14, 15]. Even simple one-dimensional flocking algorithms, like the one we use in this paper, are beyond state-of-the-art automatic verification techniques [13], let alone planar or three-dimensional algorithms, but efforts like the one we undertake at least have a hope of being automatically verified due to their relative simplicity and lower dimensionality. The composition of continuous spatial programs in the Proto spatial computing language was developed in [1] and is similar in spirit to our work, albeit without any formal verification guarantees.

Our work is related to the stream of research that converts high-level specifications of robotics plans into low-level feedback based control laws. More concretely, our work uses Linear Temporal Logic (LTL) framework that has been recently proposed [3, 4, 8–10, 17–20, 27, 30]. The main differences of these ideas with the present work are: 1) instead of translating to feedback based controllers, high-level plans are translated to verified low-level primitives whose sensing and communication requirements are reduced; and 2) the propositions are made over a discretization of the configuration space for a group of robots instead of over regions in the robots' workspace.

The methodology and contributions of our work include: (a) specifying behaviors with linear temporal logic (LTL) and synthesizing high (swarm)-level behaviors for set points of verified lower-level primitives, (b) extending a previously verified one-dimensional distributed flocking algorithm [14, 15] for creating planar formations by combining it with homogeneous transformations (rotations and translations) and distributed averaging algorithms that require mostly local information, (c) an extension of the previously verified one-dimensional safety and progress properties to planar safety and progress properties, and (d) a first step toward synthesizing verified distributed algorithms for forming "virtual" kinematic chains of mobile robots.

II. Low-Level Formation Primitives

Notation. The swarm robot system consists of N robots, each with a unique identifier in the set $[\mathsf{N}] \triangleq \{1, \ldots, \mathsf{N}\}$. Identifiers name robots, and the special symbol $\perp$ means no robot. Subscripts refer to individual robots' state components, e.g., x_i is robot i's position. For a set S, let $S_\perp \triangleq S \cup \{\perp\}$.

One-Dimensional Flocking Primitive. Each robot has a position at some discrete time on the real-line. The one-dimensional position of robot i at time step k is $x_i[k] : \mathbb{N} \rightarrow \mathbb{R}$. We assume that one robot is positioned at the origin, and all other robots are positioned toward positive infinity. The *leader* is the robot with the position at the origin, the *tail* is the robot with greatest position, and the other robots are *interior*. For robot i, let $L(i)$ be the robot with position immediately left of i's, that is, ($\{j \in [\mathsf{N}] \mid j \neq i \wedge x_i[k] > x_j[k] \wedge |x_i[k] - x_j[k]|$ is smallest$\}$ or $\perp$ if none), and let $R(i)$ be the robot with position immediately right

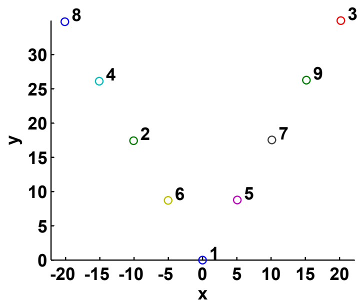

Figure 1: Vee-formation example consisting of two groups for $\mathsf{N} = 9$ robots, with robot 1 as a shared leader. The two groups are: $H = \{1, 3, 5, 7, 9\}$ and $G = \{1, 2, 4, 6, 8\}$, where 1 is the leader, 8 is G's tail, 3 is H's tail, and all other robots are interior. Video: http://www.youtube.com/watch?v=Znt3qSzGnN4.

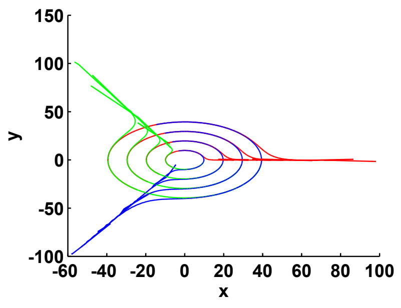

Figure 2: Y-formation and rotation for $\mathsf{N} = 13$ robots in three groups that each form a one-dimensional flock while rotating (see Figure 3), each group respectively situated along the lines with $\theta = 0$, $\theta = \frac{2\pi}{3}$, and $\theta = \frac{4\pi}{3}$. Video: http://www.youtube.com/watch?v=zeaFQ0zAk9U.

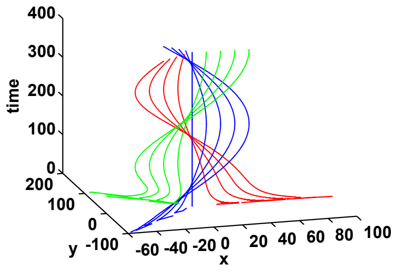

Figure 3: Y-formation and rotation positions with respect to time for $\mathsf{N} = 13$ robots in three groups. The leader's position is the origin. In the simulation, we add sensing noise to illustrate the robustness of the algorithms, due to their exponential stability. Video: http://www.youtube.com/watch?v=zeaFQ0zAk9U.

of i's, that is, $(\{j \in [\mathsf{N}] \mid j \neq i \wedge x_i[k] < x_j[k] \wedge |x_i[k] - x_j[k]| \text{ is smallest}\}$ or $\perp$ if none$)$. Each robot updates its position using the following method:

$$x_i[k+1] := \begin{cases} x_i[k] & \text{if } i \text{ is leader} \\ \frac{x_{L(i)}[k] + x_{R(i)}[k]}{2} & \text{if } i \text{ is interior} \\ \frac{x_{L(i)}[k] + x_i[k] + r_f}{2} & \text{if } i \text{ is tail} \end{cases}, \tag{1}$$

where $x_{L(i)}[k]$ is the position of the robot left of i at time k—which exists for all robots except the leader—and $x_{R(i)}[k]$ is the position of the robot right of i at time k—which exists for all non-tail robots. Note that the algorithm depends only on local information: robot i needs at most information from its left and right neighbors to update its position. Furthermore, while we assume robots update positions in a global coordinate system, the intuitive update of Equation 1 for an interior robot i is: "go to the midpoint of your nearest neighbors." It can be shown [15, Invariant 3.2] that this simple algorithm maintains safety, specified as:

$$\zeta_{\mathsf{N}}(k) \triangleq \forall i, j \in [\mathsf{N}] \ : \ i \neq j \Rightarrow |x_i[k] - x_j[k]| \geq r_s.$$

Safety holds if all robots always maintain a minimum real distance $r_s > 0$ between one another, so no two robots on the line collide, specified formally as $\forall k \in \mathbb{N}$, we have $\zeta_{\mathsf{N}}(k)$. In higher dimensions, safety is defined using an appropriate higher-dimensional norm for $|\cdot|$, e.g., the 2-norm $\|\cdot\|$. Furthermore, it can be shown [15, Theorem 3.3] that the algorithm stabilizes exponentially to a flock, defined as states where each adjacent robot is spaced apart by r_f, for some real $r_f > r_s > 0$:

$$\phi_{\mathsf{N}}(k) \triangleq \forall i \in [\mathsf{N}] \setminus \{leader\} \ : \ \left|x_i[k] - x_{L(i)}[k]\right| = r_f.$$

Two-Dimensional Formations with One-Dimensional Flocking

Using the one-dimensional update from Equation 1, we next show how to create and maintain two-dimensional formations (e.g., as shown in Figures 1 and 3). At time k, let $\mathbf{x_i}[k] = (x_i[k], y_i[k])$ be the position of robot i in the plane, where $x_i[k]$ is i's x-coordinate and $y_i[k]$ is i's y-coordinate.

For vee formation, we use two one-dimensional flocking updates, one for each of two groups of robots along two rays composing the vee-shape, which share a common leader at the vee's vertex (see Figure 1). The one-dimensional flocking update is performed in a virtual rotated reference frame. We denote the x-axis in the virtual reference frame as the $\hat{x}$-axis, and the y-axis in the rotated reference frame as the $\hat{y}$-axis. The rotated reference frame is one where the robots are situated along an axis, and we pick the $\hat{x}$-axis arbitrarily, but it could be the $\hat{y}$-axis or even an offset from another group.

The robots update their positions using the one-dimensional flocking method of Equation 1, then rotate their positions back to the original reference frame to yield a position update (control) in the original reference frame. That is, we solve the one-dimensional flocking problem in a rotated reference frame to determine a position update, then transform this updated position back into the original coordinate system. We assume the robots are roughly along the vee initially, so that left and right neighbors may be uniquely identified. For example, in Figure 1, for $i = 6$, $L(i) = 1$ and $R(i) = 2$, and earlier definitions of $L(\cdot)$ and $R(\cdot)$ in terms of $x_i[k]$ and $x_j[k]$ now use $\hat{x}_i[k]$ and $\hat{x}_j[k]$.

The algorithm is summarized by the following local sequential steps executed by each robot $i \in [\mathsf{N}]$:

(a) Rotate $\mathbf{x_i}[k]$ to a point $\hat{\mathbf{x}}_\mathbf{i}[k]$ in a virtual rotated reference frame (roughly on the $\hat{x}$-axis),

(b) Execute the one-dimensional flocking algorithm in the rotated reference frame for the $\hat{x}_i[k]$ position (Equation 2 , which is Equation 1 in the rotated reference frame),

(c) Execute a one-dimensional distributed averaging update for the $\hat{y}_i[k]$ position (Equation 3), and

(d) Rotate the virtual reference frame back to the original physical frame and update the $\mathbf{x_i}[k]$ positions.

Each robot needs primarily local sensing and communication information: the ability to estimate distance from its left and right neighbors (if any), an ability to estimate a distance r_f away (for the tail only), the position of the leader (for movement only, described later), and an ability to estimate the angles of groups oriented along different lines (e.g., the two lines making the vee).

Let G and H partition $[\mathsf{N}]$, except each has one identifier in common, that of the leader robot, which could be selected using a distributed leader election algorithm [23]. We refer G and H as *groups* of robots, and extend the definitions of leader, interior, and tail robots to refer to corresponding robots in each group. For example, in Figure 1, 1 is the leader, G contains the leader and robots with even identifiers less than $\mathsf{N} = 9$, and H contains the leader and robots with odd identifiers. For each robot $i \in [\mathsf{N}]$, let its rotational angle be denoted by $\theta_i[k] \in [0, 2\pi)$, which is the angle offset from the x-axis of the line on which robot i resides. Suppose robots in the same group are roughly on the same line, so for every $i, j \in G$, $\min\left(|\theta_i[k] - \theta_j[k]|, 2\pi - |\theta_i[k] - \theta_j[k]|\right) \leq \epsilon$ for some small real $\epsilon > 0$, and likewise for H. This assumption is easy to satisfy, using, e.g., a distributed consensus (averaging) algorithm on the angle [6, 29]. For each robot $i \in [\mathsf{N}]$, its two-dimensional position in the rotated reference frame at time k is denoted by $\hat{\mathbf{x}}_\mathbf{i}[k] \in \mathbb{R}^2$, for $\hat{\mathbf{x}}_\mathbf{i}[k] = A_i^\theta[k]\mathbf{x_i}[k]$, where:

$$A_i^\theta[k] = \begin{bmatrix} \cos\theta_i[k] & -\sin\theta_i[k] \\ \sin\theta_i[k] & \cos\theta_i[k] \end{bmatrix}.$$

Figure 1 shows an example where the rotation angles for G and H are $\frac{\pi}{4}$ and $\frac{3\pi}{4}$ off the x-axis, respectively.

In the rotated reference frame, the $\hat{x}$-axis position of each robot is updated according to the one-dimensional method (Equation 1), so:

$$\hat{x}_i[k+1] := \begin{cases} \hat{x}_i[k] & \text{if } i \text{ is leader} \\ \frac{\hat{x}_{L(i)}[k] + \hat{x}_{R(i)}[k]}{2} & \text{if } i \text{ is interior} \\ \frac{\hat{x}_{L(i)}[k] + \hat{x}_i[k] + r_f}{2} & \text{if } i \text{ is tail} \end{cases} \cdot \tag{2}$$

Additionally, each robot i updates its $\hat{y}$-position in the rotated reference frame as:

$$\hat{y}_i[k+1] := \begin{cases} \hat{y}_i[k] & \text{if } i \text{ is leader} \\ \frac{\hat{y}_{L(i)}[k] + \hat{y}_{R(i)}[k]}{2} & \text{if } i \text{ is interior} \\ \frac{\hat{y}_{L(i)}[k] + \hat{y}_i[k]}{2} & \text{if } i \text{ is tail} \end{cases}, \tag{3}$$

so that each robot's position converges to the average of their original positions in the rotated reference frame ($\hat{y}_i[k+1] \to 0$). We note that this does not cause the $\hat{y}_i[k]$ to converge to zero, so there is an offset from the axis. An alternative is to update $\hat{y}_i[k+1] = \frac{1}{2}\hat{y}_i[k]$, which would cause the robots to converge to the $\hat{x}$-axis in the rotated reference frame, but this requires that robot i know its distance from the axes (rather than just local information). This update, along with the previous flocking algorithm, are special cases of general distributed averaging algorithms, and have exponential stability so long as connectivity is maintained and are independent of which specific robots are used in the averaging [6, 25, 29]. Finally, rotate back to the original reference frame to determine the new position of robot i:

$$\begin{bmatrix} x_i[k+1] \\ y_i[k+1] \end{bmatrix} = A_i^{-\theta}[k] \begin{bmatrix} \hat{x}_i[k+1] \\ \hat{y}_i[k+1] \end{bmatrix},$$

where $A_i^{-\theta}[k]$ is the inverse rotation of $A_i^{\theta}[k]$.

III. General Multi-Group Formations

Next, we generalize the notion of groups from 2 as in the vee-formation case to 3 for y-formation (see Figure 3) and beyond. The one-dimensional algorithm each robot uses is the same as in the vee formation described in Equation 2. Let $\mathcal{G}$ be a set that partitions $[\mathsf{N}]$, except with a common leader (as in G and H in the vee formation), and we refer to $\mathcal{G}$ as the set of groups, and each $G \in \mathcal{G}$ as a group. The robots in each group in $\mathcal{G}$ perform the same local operations described as in the vee case, and the only difference now is that each group has its own distinct rotation angle θ.

Rotated Planar Safety and Progress to Flocking. The definitions of safety and progress to flocking are updated to the rotated reference frame as follows for planar formations in groups. A flock is defined the same as in the one-dimensional case, except restricted to groups, so, for a group $G \in \mathcal{G}$:

$$\phi(G,k) \triangleq \forall i \in G \setminus \{leader\} \ : \ \left|\hat{x}_i[k] - \hat{x}_{L(i)}[k]\right| = r_f,$$

which specifies that all robots in group G are spaced apart by r_f, for some real $r_f > r_s > 0$. Let planar flocking be specified as:

$$\phi_{\mathsf{N}}(k) \triangleq \forall G \in \mathcal{G} \ : \ \phi(G,k),$$

which is where all the $\mathcal{G}$ groups are in one-dimensional flocks. Each of the flocking and averaging algorithms executed by the robots are exponentially stable. Since each group $G \in \mathcal{G}$ converges exponentially to states where $\phi(G)$ is satisfied, the robots converge exponentially to states where $\phi(\mathsf{N})$ holds using our method.

Next, we update the one-dimensional version of safety to groups, and also discuss more general notions of planar safety. For a group $G \in \mathcal{G}$, let:

$$\zeta(G,k) \triangleq \forall i,j \in G \ : \ i \neq j \Rightarrow |\hat{x}_i[k] - \hat{x}_j[k]| \geq r_s,$$

for some $r_s > 0$, and safety holds for group $G \in \mathcal{G}$, if for all $k \in \mathbb{N}$, we have $\zeta(G,k)$. Safety of the entire swarm is specified as all groups being safe:

$$\zeta_{\mathsf{N}}(k) \triangleq \forall G \in \mathcal{G} \ : \zeta(G,k),$$

and safety holds for the swarm if for all $k \in \mathbb{N}$, we have $\zeta_{\mathsf{N}}(k)$. Since each group's one-dimensional flocking algorithm maintains safety, all the composition will maintain this notion of safety.

General Planar Safety and Progress to Flocking. However, better definitions of flocking and safety compare the planar distance of each robot in the plane. For a group $G \in \mathcal{G}$, let:

$$\phi(G,k) \triangleq \forall i \in G \setminus \{leader\} \ : \ \left\|\mathbf{x_i}[k] - \mathbf{x_{L(i)}}[k]\right\| = r_f,$$
$$\zeta(G,k) \triangleq \forall i,j \in G \ : \ i \neq j \Rightarrow \|\mathbf{x_i}[k] - \mathbf{x_j}[k]\| \geq r_s,$$

respectively specify flocking and safety for group G, where $\|\cdot\|$ is the 2-norm. Flocking and safety for the entire swarm are specified respectively as:

$$\phi_{\mathsf{N}}(k) \triangleq \forall G \in \mathcal{G} \ : \phi(G,k) \text{ and}$$
$$\zeta_{\mathsf{N}}(k) \triangleq \forall G \in \mathcal{G} \ : \zeta(G,k).$$

If the difference in rotation angles θ for robots in different groups is sufficiently large—i.e., $\min\left(|\theta_i[k] - \theta_j[k]|, 2\pi - |\theta_i[k] - \theta_j[k]|\right) \geq \delta$ for $i \in G_1$, $j \in G_2$ where $G_1 \neq G_2$, and for $\delta > 0$—the swarm maintains $\zeta_{\mathsf{N}}(k)$. This

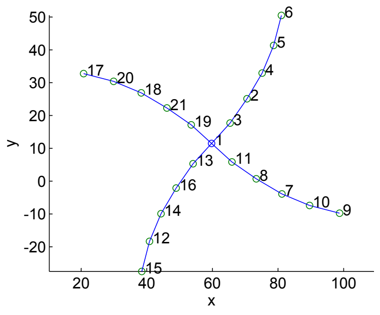

Figure 4: Formation and rotation for $\mathsf{N} = 21$ robots in four groups, that each form a one-dimensional flock while rotating. Video: https://www.youtube.com/watch?v=LtI0GiyEhQw.

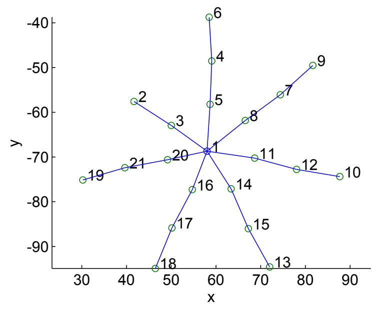

Figure 5: Formation and rotation for $\mathsf{N} = 21$ robots in seven groups, that each form a one-dimensional flock while rotating (and translating). Video: https://www.youtube.com/watch?v=0MX2iZOBMLE.

avoids scenarios where different groups have similar θ values where safety would not be maintained, so an additional distributed mutual exclusion algorithm can ensure robots in different groups do not utilize similar values of θ. In Section IV, we will illustrate a way to synthesize high-level behaviors from safety specifications that ensure this mutual exclusion property on choices of θ. For flocking, the same restriction in rotation angles is sufficient, along with the requirement that the initial positions of robots' have an average $\hat{y}$ position in the rotated reference frame of 0. This ensures the distributed averaging algorithm from Equation 3 converges to $\hat{y}_i[k] = 0$, so that the robots' positions in the rotated reference frame lie along the $\hat{y}$-axis.

While we present simple two (vee) and three (y) group formations in Figures 1, 2, and 3, the general framework is extensible to arbitrary numbers of groups. For example, we have conducted simulations (in our Matlab simulator) with up-to tens of groups. Figure 4 shows a rotation scenario with four groups and $\mathsf{N} = 21$ robots, and Figure 5 shows a rotation scenario with seven groups and $\mathsf{N} = 21$ robots. Additionally, more general "virtual" open kinematic chains may be created by generalizing the notions of groups but using basic homogenous transformations that make up the flocking algorithm and verified primitives [21].

Swarm Rotational and Translational Movements. Rotational movement of robots in a group G may be accomplished by adding the same small constant $\delta > 0$ to $\theta_i[k]$ for each robot $i \in G$ at each time step:

$$\theta_i[k+1] = \theta_i[k] + \delta,$$

and update $A_i^{\theta}[k+1] = A_i^{\theta}[k]$. Another distributed consensus algorithm can be run to decide a common δ. If δ is small and the same for all robots in all groups, then $\zeta(\mathsf{N})$ and $\phi(\mathsf{N})$ are both maintained. This is also what we observed in our simulations, illustrated in Figure 3. Due to space, we omit the details of performing translations of the swarms, but the non-leader robots basically subtract off the leader's position in the rotated reference frame (so the leader is situated at the origin), perform the one-dimensional flocking, then add back the leader's position. Overall, the operations of the rotation and translation are standard homogeneous transformations [21].

IV. High-Level Synthesis for Low-Level Primitives

Such two dimensional and more complex formations of two or more subgroups can be generated more naturally with a high-level specification language like linear temporal logic (LTL). High-level LTL specifications determine the behavior of a group of agents. Each agent is a member in one of several one-dimensional flocks (groups). This membership implies a set of rules that utilize minimal information, avoid inter-agent collisions, and have exponential stability (for example under consensus these networks will remain connected and converge to their centroid). Alone, however, these flocks do not engage in especially interesting or complex patterns, although one-dimensional flocks are useful for modeling phenomena like platooning. Thus, we will specify more complex behavior at a higher level of abstraction using LTL as in [16, 22], and leave collision avoidance, etc. to a verified low-level flocking algorithm such as [15]. This is in line with previous attempts at organizing robotic behavior [5], generating more complex movements from simpler ones [22], and creating unusual multi-agent patterns [28]. This section explains the use of LTL for composing these flocking primitives, establishes a transition system model, and provides example specifications.

Notation. The spacing factor for each flock will be variable between the minimal r_s to two preset values $r_{lo} > r_{ti} > r_s$. The angle of the flock off the leader with respect to the x-axis will vary between three presets as well, $\theta_0 < \theta_{ac} < \theta_{ob}$.

Representation via a Labeled Transition System. The behavior of i one-dimensional flocks may be formally modeled as a labeled transition system (see Figure 6): $T = (Q, q_0, \rightarrow, \Pi, h)$, where Q is a set of states with initial state q_0, $\rightarrow$ indicates the transitions, Π is the set of atomic propositions, and h labels each state with the appropriate proposition as in [22]. The states of T correspond to manipulatable primitive parameters (r_f, θ) and are labeled with the propositions dealing with descriptions of the resulting formations given in Equation 4. Our low-level flocking behaviors ensure the system is constantly switching between distinct formations out of some set and the structure of this transition system will describe the dynamics of this process. In Figure 6, descriptive, high-level words associated with each formation are used along with other words that might be natural to frame a specification around. Key formations of interest are enumerated for illustrative purposes, labeled F_1, F_2, F_3, and pictured in Figure 7.

To make more complex formations, with multiple primitives, this primary transition system may be composed with others. For example, the transition system governing the behavior of two flocking primitives would be given by $T_1 \otimes T_2$ with $(Q_1 \times Q_2, q_{01} \times q_{02}, \rightarrow_P, \Pi_1 \cup \Pi_2, h_P)$. The new transition function $\rightarrow_P$ is defined if and only if a transition existed between both single states, i.e. $(q, q') \in \rightarrow_P$ if and only if $q \neq q'$, $(q_1, q_1') \in \rightarrow_1$ and $(q_2, q_2') \in \rightarrow_2$, where $q = (q_1, q_2)$ and $q' = (q_1', q_2')$. The labeling function h_P associates any proposition that was true for either single leg state with the joint state, i.e. $h_P : Q_1 \times Q_2 \mapsto 2^{\Pi_1 \cup \Pi_2}$.

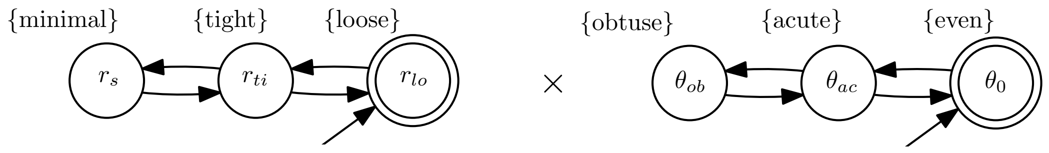

Figure 6: The transition system governing the behavior of a single primary flock is given in the figure above as the Cartesian product of two simpler machines. Each state is augmented with a set of atomic propositions over which a specification may act and their atomic propositions (labels).

High-Level Control via Formal Language Specification

A high-level controller is constructed for the transition system such as to impart rules—which may be style based—and to generate more diverse behavior. The atomic propositions are statements which are either true or false about every state of our system. We use temporal logic to watch the evolution of our system in terms of these statements of particular interest; hence, we often think of these propositions as observations.

We use the following notation for our LTL formulas (see [16, 22] for more detail): a set of atomic propositions Π, Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (implication) and temporal operators **X** (next), $\mathcal{U}$ (until), **F** (eventually), **G** (always). The semantics of LTL formulas are given over infinite words generated by a labeled transition system (T); states of the transition system are labeled with atomic propositions. We can then watch T evolving according to these propositions: that is, a word of T corresponds to a (infinite) sequence of sets of the propositions in Π.

For the flocking primitives outlined in the previous section, we will use the transition system in Figure 6, where we consider the state of our system to be based on a spacing parameter and angle of orientation of the leader agent, and name the following set of atomic propositions:

$$\Pi = \{\text{minimal, tight, loose, even, offset, orthogonal}\}, \tag{4}$$

where “minimal,” “tight,” and “loose,” refer to the value of a spacing factor r_f and “even,” “offset,” and “orthogonal” refer to the value of the angle of the leader agent. From these descriptive parameters, we can name specific formations and generate specifications to see that these formations are achieved in the desired way. A subscript of 1 or 2 will be appended to indicate to which flocking primitive each proposition refers.

Safety Specification. First, universal specifications regarding the safe inter-operation of the individual flocks will be enumerated. These two specifications will govern the system simultaneously with additional specifications in all formations constructed in Section III. First, a specification is needed to ensure that the two (or more) composed flocks are not oriented in the same way, which would be equivalent to requesting the groups to operate in the same space. The second specification ensures that each flock operates with the same spacing parameter. This consideration may be necessary in certain environments or at certain speeds of the agents, where greater or lesser spacing may be required. These two specifications are written in this framework as:

1. Request that the system never uses the same orientation for composed flocks.
   Never use the same orientation as subgroup 1:
   $\phi_{s1} = \mathbf{G}[(\text{even}_1 \rightarrow \neg(\text{even}_2)) \wedge (\text{acute}_1 \rightarrow \neg(\text{acute}_2)) \wedge (\text{obtuse}_1 \rightarrow \neg(\text{obtuse}_2))]$.

2. Request that the system always uses the same spacing for composed flocks.
   Always use the spacing of subgroup 1:
   $\phi_{s2} = \mathbf{G}[(\text{minimal}_1 \rightarrow \text{minimal}_2) \wedge (\text{tight}_1 \rightarrow \text{tight}_2) \wedge (\text{loose}_1 \rightarrow \text{loose}_2)]$.

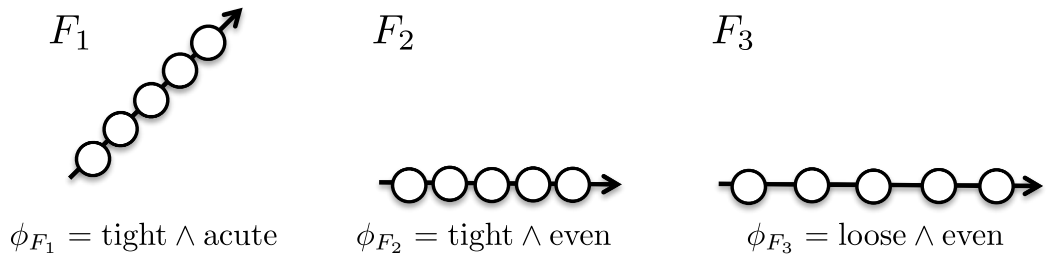
F_1
F_2
F_3
$\phi_{F_1} = \text{tight} \wedge \text{acute}$
$\phi_{F_2} = \text{tight} \wedge \text{even}$
$\phi_{F_3} = \text{loose} \wedge \text{even}$

Figure 7: Key formations of interest, described by an appropriate LTL formula.

Formation Control. In order to create formations of interest such as those shown in Figure 7, particular configurations may be denoted or specified with their own LTL formula as shown in Figure 8. Then, additional specifications may be phrased as follows.

1. Request that the system always returns to Formation 2: $\phi_{f1} = \mathbf{G}\ \mathbf{F}\ \phi_{F_2}$.

2. Request that the system never enter Formation 3 after Formation 1: $\phi_{f2} = \phi_{F_1} \wedge \mathbf{X}\neg(\phi_{F_3})$.

3. Request that the system form a vee-formation: $\phi_{f3} = \mathbf{G}\ [(\text{acute}_1 \wedge \text{obtuse}_2) \vee (\text{obtuse}_1 \wedge \text{acute}_2)]$.

From these four formula, two final specifications may be concatenated:

1. Request that the system always returns to Formation 2 and adhere to safety considerations: $\phi_1 = \phi_{f1} \wedge \phi_{s1} \wedge \phi_{s2}$.

2. Request that the system never enter Formation 3 after Formation 1 and adhere to safety considerations: $\phi_2 = \phi_{f2} \wedge \phi_{s1} \wedge \phi_{s2}$.

3. Request that the system achieve vee-formation and adhere to safety considerations: $\phi_3 = \phi_{f3} \wedge \phi_{s1} \wedge \phi_{s2}$.

From each of the final specifications a Büchi automaton, $B_\phi = (S, S_0, \Sigma = 2^\Pi, \delta, F)$, is constructed as in [22]. These automata were constructed using LTL2BA [11] and are shown in Figure 8. Then, the automata are composed with the system transition model, $T_1 \otimes T_2$. This final system allows only sequences in line with the original system dynamic and the structure of atomic propositions as governed by the specifications. Formally, it is given as $\mathcal{A} = (T_1 \otimes T_2) \times B_\phi = (Q \times S, q_0 \times S_0, \delta_A, F_A)$ where $(q_j, sl) \in \delta_A((q_i, s_k))$ iff $(q_i, q_j) \in \rightarrow$ and $s_l \in \delta(s_k, h(q_i))$. In other words, $\mathcal{A}$ encodes the system dynamics of T and the specification contained in ϕ. Another way of thinking of it is that the Büchi automaton generated by ϕ is a kind of high-level controller for the system T. Acceptable runs of $\mathcal{A}$ are thought of as output behavior.

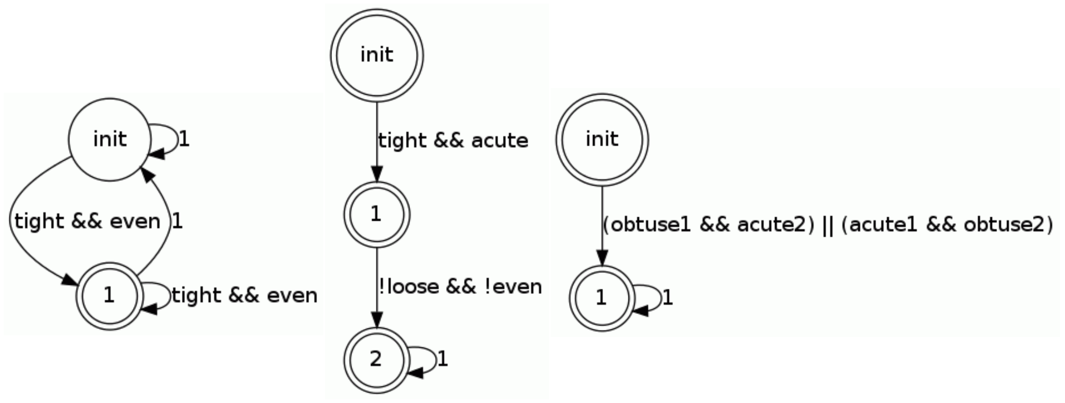

Figure 8: Büchi automatongiven by the specifications; these automata act as supervisors for $T_1 \times T_2$ for ϕ_1, ϕ_2, and ϕ_3, left to right.

Satisfying Example Runs. Under the final specification ϕ_1, an accepted sample of system behavior is given by:

$$\begin{aligned}&\{(\text{Minimal}_1, \text{Obtuse}_1), (\text{Minimal}_2, \text{Acute}_2)\}, \{(\text{Minimal}_1, \text{Acute}_1), (\text{Minimal}_2, \text{Obtuse}_2)\},\\ &\{(\text{Minimal}_1, \text{Even}_1), (\text{Minimal}_2, \text{Obtuse}_2)\}, \{(\text{Tight}_1, \text{Acute}_1), (\text{Tight}_2, \text{Obtuse}_2)\},\\ &\{(\text{Tight}_1, \text{Acute}_1), (\text{Tight}_2, \text{Obtuse}_2)\}, \{(\text{Tight}_1, \text{Even}_1), (\text{Tight}_2, \text{Acute}_2)\}, ...\end{aligned} \tag{5}$$

Under the final specification ϕ_2, an acceptable sample of system behavior is given by:

$$\begin{aligned}&\{(\text{Minimal}_1, \text{Obtuse}_1, (\text{Minimal}_2, \text{Acute}_2)\}, \{(\text{Tight}_1, \text{Obtuse}_1), (\text{Tight}_2, \text{Even}_2)\},\\ &\{(\text{Loose}_1, \text{Acute}_1), (\text{Loose}_2, \text{Even}_2)\}, \{(\text{Tight}_1, \text{Acute}_1), (\text{Tight}_2, \text{Even}_2)\},\\ &\{(\text{Loose}_1, \text{Obtuse}_1), (\text{Loose}_2, \text{Acute}_2)\}....\end{aligned} \tag{6}$$

Under the final specification ϕ_3, an accepted sample of system behavior is given by:

$$\begin{aligned}&(\text{Minimal}_1, \text{Obtuse}_1), (\text{Minimal}_2, \text{Acute}_2), (\text{Minimal}_1, \text{Acute}_1), (\text{Minimal}_2, \text{Obtuse}_2),\\ &(\text{Tight}_1, \text{Acute}_1), (\text{Tight}_2, \text{Obtuse}_2), (\text{Loose}_1, \text{Acute}_1), (\text{Loose}_2, \text{Obtuse}_2)....\end{aligned} \tag{7}$$

This example shows how the work in the previous section may be achieved in a more natural way with this framework – made possible by the primitive formalism. More complex formations depend on further generalizations of the flocking primitives. For example, one state may be where the agents form a more general kinematic chain structure, or a circle and it may have the labels 'circle,' 'concave,' and 'convex.'

V. Conclusion and Future Work

We present a methodology for synthesizing control actions for verified low-level primitives using higher-level formal specifications in LTL. Specifically, we present a simple method to combine one-dimensional distributed flocking/platooning and averaging protocols—that have some verified properties like safety (collision avoidance) and exponential stability—to generate complex higher-dimensional behaviors, such as vee-formation, y-formation, and movements and rotations thereof. Beneficial properties of the underlying primitives—such as exponential convergence—allow robustness of the higher-dimensional behaviors to disturbances like noise, and are able to guarantee safety under some reasonable assumptions about overlapping angles. We also plan to study a more formal proof of the exponential stability of the composed system, perhaps by using methods for composing stability proofs [24]. Additionally, while we present our approach for planar formations, the compositional approach also can be used for constructing three-dimensional formations (e.g., using yaw, pitch, and roll).

References

[1]J. Bachrach, J. Beal, and J. McLurkin. Composable continuous-space programs for robotic swarms. *Neural Computing & Applications*, 19:825–847, 2010.

[2]T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[3]A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based motion planning with temporal goals. pages 2689–2696, 2010.

[4]L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. M. LaValle. Controlling wild bodies using linear temporal logic. *Robotics: Science and Systems VII*, page 17, 2012.

[5]R. A. Brooks. Behavior-based humanoid robotics. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 1, pages 1–8. IEEE, 1996.

[6]F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks.* Applied Mathematics Series. Princeton University Press, 2009. To appear. Electronically available at http://coordinationbook.info.

[7]M. Egerstedt and X. Hu. Formation constrained multi-agent control. *Robotics and Automation, IEEE Transactions on*, 17(6):947–951, Dec. 2001.

[8]G. E. Fainekos. Revising temporal logic specifications for motion planning. 2011.

[9]G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[10]C. Finucane, G. Jing, and H. Kress-Gazit. Ltlmop: Experimenting with language, temporal logic and robot control. In *IROS*, pages 1988–1993, 2010.

[11]P. Gastin and D. Oddoux. Fast LTL to Buchi automata translation. *Lecture Notes in Computer Science*, pages 53–65, 2001.

[12]V. Gazi and K. M. Passino. Stability of a one-dimensional discrete-time asynchronous swarm. *IEEE Trans. Syst., Man, Cybern. B*, 35(4):834–841, Aug. 2005.

[13]T. T. Johnson. *Uniform Verification of Safety for Parameterized Networks of Hybrid Automata.* PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL 61801, 2013.

[14]T. T. Johnson and S. Mitra. Safe flocking in spite of actuator faults. In S. Dolev, J. Cobb, M. Fischer, and M. Yung, editors, *12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2010)*, volume 6366 of *Lecture Notes in Computer Science*, pages 588–602. Springer Berlin / Heidelberg, Sept. 2010.

[15]T. T. Johnson and S. Mitra. Safe flocking in spite of actuator faults using directional failure detectors. *Journal of Nonlinear Systems and Applications*, 2(1-2):73–95, Apr. 2011.

[16]M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1):287–297, 2008.

[17]M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics and Automation*, 26(1):48–61, 2010.

[18]H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, Oct. 2008.

[19]H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics and Automation*, 25(6):1370–1381, Dec. 2009.

[20]M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *ICRA*, pages 3227–3232, 2010.

[21]S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, U.K., 2006.

[22]A. LaViers, Y. Chen, C. Belta, and M. Egerstedt. Automatic sequencing of ballet poses. *Robotics and Automation Magazine, IEEE*, 18(3):87–95, 2011.

[23]N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[24]C. Mitrohin and A. Podelski. Composing stability proofs for hybrid systems. In U. Fahrenberg and S. Tripakis, editors, *Formal Modeling and Analysis of Timed Systems*, volume 6919 of *Lecture Notes in Computer Science*, pages 286–300. Springer Berlin Heidelberg, 2011.

[25]R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007.

[26]C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM.

[27]S. L. Smith, J. Tumova, C. Belta, and D. Rus. Optimal path planning under temporal logic constraints. In *IROS*, pages 3288–3293, 2010.

[28]P. Tsiotras and L. R. Castro. Extended multi-agent consensus protocols for the generation of geometric patterns in the plane. In *American Control Conference (ACC)*, 2011.

[29]J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Trans. Autom. Control*, 31(9):803–812, Sept. 1986.

[30]M. Wu, G. Yan, Z. Lin, and Y. Lan. Synthesis of output feedback control for motion planning based on ltl specifications. In *IROS*, pages 5071–5075, 2009.