$$xx(1.96 \pm 1)$$

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

# Un algoritmo de auto-organización para la formación de agentes móviles



Tesis que presenta:

**Miguel Angel Sánchez Acevedo**

para obtener el grado de:

**Maestro en Ciencias**

en la especialidad de:

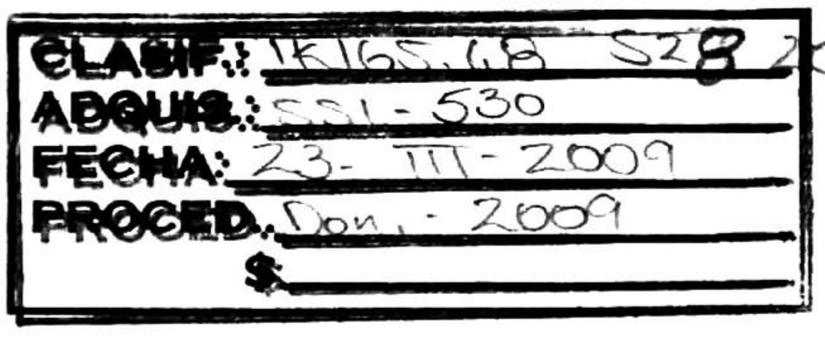
**Ingeniería Eléctrica**

Directores de Tesis

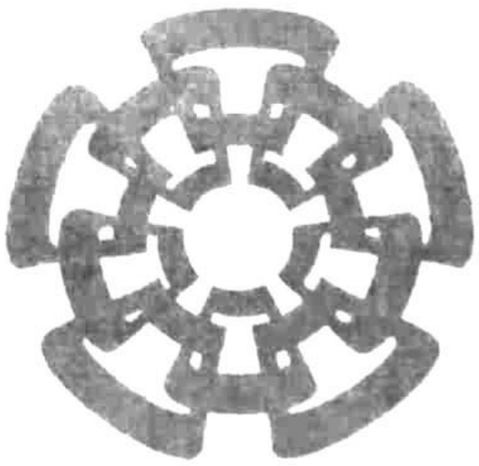
**Dr. Luis Ernesto López Mellado**
**Dr. Félix Francisco Ramos Corchado**



Guadalajara, Jalisco, Agosto de 2008.




ID: 158264-1001



Centro de Investigación y de Estudios Avanzados

del I.P.N.

Unidad Guadalajara

# A Self-organization Algorithm for Mobile Agents Formation

A thesis presented by:

**Miguel Angel Sánchez Acevedo**

to obtain the degree of:
**Master of Science**

in the subject of:
**Electrical Engineering**

Thesis Advisors:
**Dr. Luis Ernesto López Mellado**
**Dr. Félix Francisco Ramos Corchado**

Guadalajara, Jalisco, August 2008.

# Un algoritmo de auto-organización para la formación de agentes móviles

**Tesis de Maestría en Ciencias**
**Ingeniería Eléctrica**


Por:

**Miguel Angel Sánchez Acevedo**

Ingeniero en Sistemas Computacionales

Instituto Tecnológico de Tehuacán 2000-2004



Becario de Conacyt, expediente no. 203124


Directores de Tesis

**Dr. Luis Ernesto López Mellado**

**Dr. Félix Francisco Ramos Corchado**


CINVESTAV del IPN Unidad Guadalajara, Agosto de 2008.

# A Self-organization Algorithm for Mobile Agents Formation

**Master of Science Thesis**
**In Electrical Engineering**


By:
**Miguel Angel Sánchez Acevedo**
Engineer in Computer Science
Instituto Tecnológico de Tehuacán 2000-2004



Scholarship granted by CONACYT, No. 203124



Thesis Advisors:
**Dr. Luis Ernesto López Mellado**
**Dr. Félix Francisco Ramos Corchado**



CINVESTAV del IPN Unidad Guadalajara, August, 2008.

# Resumen

Esta tesis presenta un algoritmo para el establecimiento y mantenimiento de una formación de agentes móviles. El algoritmo propuesto está basado en auto-organización, un fenómeno observado en la naturaleza donde la organización del sistema surge de las interacciones locales entre los componentes del mismo. Se considera la auto-organización debido a sus propiedades inherentes: robustez, escalabilidad y adaptabilidad.

Con el fin de establecer una formación de agentes móviles, cada agente involucrado en el proceso de formación ejecuta el mismo algoritmo. La formación se establece como consecuencia de las interacciones entre los agentes. Con el algoritmo propuesto en esta tesis se pueden obtener tres formaciones diferentes: línea, columna y cuña. Se incluye la posibilidad de cambiar entre formaciones en tiempo de ejecución. Durante la navegación, la presencia de obstáculos puede disminuir el avance de los robots e influenciar a la desintegración de la formación, para resolver este problema se propone un esquema que puede ser implementado por los robots con el fin de evitar los obstáculos.

Para llevar a cabo la auto-organización cada agente tiene conocimiento de sus vecinos, pueden comunicarse entre ellos y las acciones realizadas por un agente provocan reacciones en sus vecinos. Estas reacciones traen como consecuencia la auto-reconfiguración de la formación; de este comportamiento se obtiene un patrón de formación. Nuevos agentes pueden ser integrados a la formación en cualquier momento. Conforme nuevos agentes se integran se lleva a cabo un comportamiento de auto-reconfiguración para mantener balanceada la formación. En una formación de cuña el agente ubicado al centro de la formación actúa como líder. Este rol es reasignado cada vez que un nuevo agente se coloca al centro de la formación.

# Abstract

This thesis presents an algorithm for establishing and maintaining a formation of mobile agents. The proposed algorithm is based on self-organization, a phenomenon observed in nature where system organization arises from local interactions between the system components. Self-organization is chosen by its inherent properties: robustness, scalability, and adaptability.

In order to establish a formation of mobile agents, every agent, which is involved in the formation process, executes the same algorithm. Formation is established as consequence of the interactions between agents. Three different formations are obtained by the schema proposed here: line, column, and wedge; also the possibility of switching among formations on the fly is included. During navigation, the presence of obstacles could decrease progress of robots and to influence the formation disintegration. To solve this problem, a simple avoiding algorithm is performed by the robots in the formation.

For improving self-organization, agents involved in a formation have knowledge about its neighbors. Neighbor agents can communicate with each other. The actions performed by an agent cause reactions on its neighbors; these reactions provoke the self-reconfiguration of the formation. A global formation pattern is obtained from this behavior. New agents can be added to the formation at anytime. When agents are joined to the formation, a self-reconfiguration behavior is accomplished for keeping a balanced formation. In a wedge formation, middle agent acts as a leader. The leader role is reassigned everytime a new agent arrives to the formation.

# Acknowledgments

I would to thank to God who gave me opportunity of ending the mastery. To my family, professors, supervisors, Conacyt, friends and classmates.

I thank my parents Tomás Sánchez Mandujano and Hortencia Acevedo Rodriguez. for their moral support and their life long advice and teach.

I also thank my professors for guiding my learning and contributing to my professional development.

Thanks to my supervisors Dr. Félix Ramos Corchado and Dr. Luis Ernesto López Mellado for sharing their ideas and putting special interest in this thesis project.

Thanks to my friends and classmates for listening to me.

Thanks to Gustavo A. Torres Blanco and Salvador Jauregui Ortiz who implemented the algorithm proposed in this thesis in NXT Robots as part of their formation in the course of Distributed Systems at CINVESTAV.

Finally, I would to thank to Conacyt for supporting my studies.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

In recent years, mobile agent formations have been widely studied [10, 14, 23, 33, 2]. Efforts for developing controllers, which allow a group of mobile agents to move from one point to another, have been increased due their applicability in several tasks. Among applications of mobile agent formations, those related with exploration, collaborative sensing, and task allocation in a group of robots, have been widely approached [6, 30, 35]. However, although several works have been proposed, they lack of scalability, robustness, or adaptability; these properties are desirables in a group of agents exploring, sensing, or working in unknown environments.

## 1.1 Problem Definition

The problems considered in this thesis in order to contribute to the state of the art of the robot formation control, include the possibility of that new elements can be added to the formation at anytime, everywhere, and the shape of the formation can be reconfigured; also, to remove the need of fixed positions for robots in formation in order to allow recovery after fails, and to include switching formation on the fly in order to robots can be adapted to the environment. Once all these problems are solved, a robust, scalable and adaptable algorithm for robot formation is obtained.

A phenomenon observed in nature called Self-organization allows that complex behaviors can be obtained through local interactions. Systems exhibiting self-organization maintain properties of scalability, robustness and adaptability. This phenomenon has been studied in many areas, namely Physics, Thermodynamics, Cybernetics, Computing modeling, Economics, and Biology [5, 16, 18]. Some results in Computing modeling obtained by Reynolds [31] show how simple rules, which control individual behavior of birds, bring as consequence the behavior performed by a flock of birds. In this work, formations of mobile agents are

obtained as a consequence of simple rules performed by individual agents.

Behavior of agents has been studied computationally with discrete dynamical systems (DDS) [19, 8]. The transitions between states are interpreted as laws or rules. These rules define the behavior of the elements of a self-organized system. Tools for modeling discrete dynamical systems such as automata, process algebra, Petri Nets among others allow to verify the correct behavior of each element in the system. However, the observed behavior of self-organization cannot be described by these transition rules. These emergent patterns pertain to a different, complementary level of observation of the same system [29].

## 1.2 Thesis objective

This work addresses the problem of establishing a formation of mobile agents. Resulting formation maintains properties of scalability, robustness, and adaptability. Self-organization principles are used for defining simple behavior rules. These behavior rules define the controller for each agent. Three formations are considered: line, column, and wedge. Leader of formation decides which formation shape is accomplished. The leader role can be played by any agent. Agent placed in the middle of a line or wedge formation is elected as leader. The element at front of a column formation plays the leader role. Computational complexity is linear increased with the number of agents.

Shifting between formations gives to agents the capability of adapting the formation in order to continue they work in changing environments. Therefore, agents in formation are arranged according to the activities they have to perform. Shifting process is initiated by the leader. Agents decide their position into formation according to the information received from their neighbors. In a wedge formation, when new agents are added, the leader verifies if a balance maneuver is required. Every time a balance maneuver is performed, the leader role is reassigned. The agents do not have knowledge about the number of agents participating in the formation.

A formal description of the algorithm is presented. Petri Nets are used for modeling the agent behavior. These formalism has been elected by the clarity of graphical states representation. Transitions between places are synchronized with external received signals. Considering these signals, behavior rules can be obtained automatically from the Petri Net model.

## 1.3 Thesis organization

This document is organized as follows:

- Chapter 2. In this chapter a breve description of the more relevant works on this area is presented.
- Chapter 3. This chapter describes the proposed algorithm and the formal description using Petri Nets.
- Chapter 4. A case of study is described in this chapter. Two simulated environments are considered. In the first one, the agent behaviors are modelled; in the second one, the algorithm is performed by simulated robots. Finally, simulation results are presented.
- Chapter 5. Conclusions obtained from this research work are presented. Finally, future work is discussed.

# Chapter 2

# Robot Formation Control

## 2.1 Introduction

Control of a group of mobile agents has received a lot of attention from the research community. This is mainly due to the variety of applications that can be accomplished by autonomous groups of mobile agents. Some examples of these applications are: distributed task assignment, exploring unknown environments, search and rescue operations, mobile sensing networks, and cooperative transportation. Formations provide a more rigid and reliable structure for agents interacting in unknown and hazardous environments. It is required that formations can be adapted to the environment as well as incorporate new elements without increasing the complexity of their controllers. Therefore, robustness, scalability, and adaptability are desirable properties for formation control.

Systems exhibiting self-organization have been widely studied due their inherent properties: robustness, scalability, and adaptability. The complex behavior of these systems is obtained from local interactions between the system components. It is necessary to identify appropriate self-organization principles for controlling the overall behavior of a group of mobile agents arranged in a formation. Advantages of improving self-organization in robot formations are: failures and unplanned behavior of individual agents not affect task completion, parallelism can be exploited, and sensing can be distributed. As consequence, the cost of constructing robots is decreased since they only perform simple tasks.

In next sections, relevant approaches proposed in this area are presented. They are organized according to the way that the problem is addressed. First, self-organization applied to multi-agent systems is described; next, controlling a formation by the system dynamics is explained; then, approaches where local information is used in order to control robot formations are studied; finally, the attempts of improving self-organization in robot formation control are introduced.

## 2.2 Self-Organization in Multi-Agent Systems

The need of controlling the behavior of agents, while scalability and robustness properties are maintained, has brought as consequence the implementation of self-organization in multi-agent systems.

In order to explain and describe how self-organization arises in multi-agent systems, Paranuk and Brueckner [28] showed through a model of pheromone-based behavior how coordination can arise. They describe the system in two levels: the macro level, which hosts self-organization, and the micro level where a random process increases the entropy of the elements. According entropy is increased at the micro level, it is reduced in the macro level. Through this simulation it could be observed that the behavior of elements at macro level depends on the entropy at micro level. A drawback of this approach is that there is no control in the shape of the formations obtained.

A model for applying self-organization in a multi-robot system was proposed in [32]. The model proposed here is called "Digital Hormone Model" (DHM). This model allows a group of mobile robots can be organized in global patterns through local interactions. The actions, which can be performed by homogeneous cells representing mobile agents, are regulated by hormones which are secreted by the cells. This model can be extended to mobile robots where every robot is able to perform actions such as migration, secretion, dead among others. The robots react to the presence of hormones according to their behavior rules; these rules determine the actions to be accomplished by the robot. Global patterns are obtained as a result of these behaviors. This work show how biological principles can be applied to computational systems in order to obtain a self-organized system. However, the formation pattern can not be controlled due the allowed actions in this work.

In earlier works, the possibility of obtaining global patterns through local interactions was introduced; however, in those works there was no control in the obtained shape. An approach presented in [24] allows a group of mobile robots self-organize in different polygon shapes. For establishing a global pattern, a barycenter is determined in the group while the rest of robots are distributed around it. In order to accomplish a regular polygon shape, many leaders are elected; furthermore, every leader is equidistant from each other. The behavior obtained by the rest of robots is the formation of lobes; these lobes give the shape of the polygon. The regular polygon shape obtained depends on the established number of leaders. A disadvantage of this work is that only polygonal shapes can be obtained.

With the idea of not only obtain regular polygon shapes, the approach proposed in [15], permits that elements involved in the formation can be arranged in spatial patterns like crystals. Elements involved in the formation generate virtual springs between their neighbors. The internal structured pattern is controlled by tuning the spring constant and the length of the spring. The values assigned to these variables are determined by trial and error

according to the desired shape and the number of elements in the formation. With this algorithm, different shapes can be obtained.However, the main drawback consist in that the tuning of the parameters is difficult and the time of convergence is increased with complex shapes.

The possibility of obtaining an arbitrary shape by a swarm of mobile agents was introduced in [9]. When an agent meets to another agent into the shape, the new agent obtains its position performing a trilateration process. Agents, which are not in the border of the shape, follow a gas expansion model; these agents are distributed uniformly into the shape due to that behavior. Every agent can perceive a faulty coordinated system, so, it is possible to recover the shape from large scale errors. Once an agent has detected a faulty coordinated system, it moves to a correct position in order to fix its coordinates. As consequence of that behavior, the rest of agents are dispersed uniformly until fill the shape. Although any shape can be obtained with this approach, a disadvantage is that some agents have to maintain a fixed position in order to keep the desired shape.

## 2.3 Robot Formation Control

### 2.3.1 Control at the Dynamics Level

One line of research for solving the problem of robot formation control is addressed at the dynamics level. Control laws are proposed for maintaining a stable formation while a trajectory tracking is performed. A feedback controller is designed for each robot.

A combination of path tracking approaches and virtual structure is used in [11] for defining the formation architecture. For establishing the formation, every robot needs to follow a reference path. The reference path of each robot is obtained from the virtual structure. A backstepping technique is applied for fixing tracking errors. The controller is designed in such a way that the derivative of the path parameter of every robot is saved as a control input. This control input is used for synchronizing all the path parameters. In the schema proposed here, the formation is maintained because every robot maintains its position into the formation; however, the position of every robot is defined a priori according to the virtual structure and the identifier of the robot. A disadvantage of this work is that if new elements need to be integrated into the formation, the virtual structure has to be modified.

Instead of manipulating a group of mobile robots by its individual elements, the approach proposed in [25] treats the group as a rigid body. A center of mass is obtained according to the number of elements in the group and the formation shape. A coordinate system based on the center of mass is described; this coordinate system determines the relative position of each robot into the formation. The center of mass is driven according to the

desired trajectory while every robot maintains its position into the formation. A singular perturbation approach is performed for decoupling the dynamics of the center of mass and the dynamics of the group shape. The center of mass is obtained according to the configuration space, which is obtained from the shape and the number of robots. This assumption has the disadvantage of new elements cannot be added on the fly and the formation cannot be changed.

A tracking controller for leader-follower schema in robot formation control is proposed in [10]. In this work a nonlinear feedback control input is defined for velocity tracking, and control laws are defined to maintain the distance with respect to the leader. Separation distances are measured from the back of the leader for avoiding collisions. The leader of the formation follows a virtual leader for trajectory tracking. Every follower needs to know the dynamics of the leader in order to provide a valid feedback control scheme. The Cartesian position and orientation of each robot are needed to obtain the kinematic controller. This work only presents the control laws to maintain a position assigned in a formation. The main drawback consist in that how the formation is established is not considered.

### 2.3.2 Control using Local Information

Another line of work addressed for solving the problem of robot formation control is based on controlling the behavior of robots through local interactions. In this line, the need of global information is reduced and in some works it is completely removed. However, in some works, the position of the robots in the formation depends on their identifiers. When this assumption is removed, the position of every robot is assigned by the leader.

A behavior-based approach to robot formation control is presented in [6]. Four formations are considered: line, wedge, column, and diamond. Every robot has a position assigned in the formation according to its identifier. To maintain the position into the formation, every robot computes its position based on the location of the other robots. Three techniques are used to determine the position: unit center referenced, leader referenced, and neighbor referenced. A line from the unit center to the next navigational point determines the orientation of the formation. The overall behavior of a robot is implemented with several motor schemes. These motor schemes allow robots to move toward a goal location while obstacles are avoided. In this work the position of the robots is fixed; this assumption makes difficult the recovery of formation after that some elements fail.

In the approach proposed in [13, 14] a group of n mobile robots are arranged in formations like column, line, diamond and wedge. Every robot has a designated neighbor to follow. One robot is established as the conductor of the formation; this robot broadcasts the information about the formation. Robots into the formation have knowledge about the number of robots participating and their identifiers. The position of every robot is determined according to

the formation shape and the robot identifier. The behavior-based controller is composed of three concurrent behaviors: channelNListener, which receives information about other robots; channelCListener, which receives information from the conductor; and the main behavior composed of the get-in-place and the look-ahead sub-behaviors, which allow robots to get their position. Although robots decide their own position into the formation, that decision is based on the number of robots participating and the identifiers; so, all the formation has to be re-organized whether new elements are joined.

One of the main problems of the works mentioned before is the fixed position of the robots into the formation. This assumption is removed in [26]. This approach maintains simple geometric formations without centralized coordination. The formation dynamically grows from single robots until a complex formation is obtained. Every robot is initialized with the shape and size of the formation to be established. The controller of each robot consists of a collection of layered behaviors; robots maintain their position into the formation through these behaviors. In a wedge formation, the leader periodically initiates a discovery protocol to determine the global state of the formation. Balance maneuver is performed if required. The approach of this thesis differs from [26] in that the agents do not need to know how many robots are in formation, and which the leader is. So, the formation algorithm proposed in this thesis is more scalable since it is not neccesary to obtain information from all the agents in the formation.

For determining autonomously the position of the robots into the formation,the approach proposed in [21] allows mobile robots to get into formation by starting a discovery phase to detect neighbor robots. The control architecture is divided in two levels: behavior level, which is made of behavior-producing modules allowing a robot to react to situations encountered in the environment; and the recommendation level, which manages the different states allowed for the control of the formation. Each robot fills an N x N matrix with the location of participating robots; the robots obtain the location of their neighbors by rotating on themselves. A depth first search is performed by each robot to identify its position in the formation. The depth first search has to be performed by every robot when new robots are added. This process has to be repeated by all the group reducing the performance of the algorithm.

Deformation of the troop is a drawback presented in robot formation control using local information. For solving this problem, an approach for maintaining the formation using behavior parameterization is presented [23]. For maintaining the formation, the next behaviors are performed by every robot: reference neighbor following, limited passivity, reaching a target position, waiting for the follower, and priority respect. Formation maintenance arises from the combination of these behaviors. The behavior accuracy is optimized by tuning behavior parameters. This parameterization is performed manually, so the parameters have to be adjusted for every formation.

### 2.3.3 Control using Self-Organization

In order to solve the problem of robot formation control in an elegant way with a simple reactive strategy, self-organization has been used for enabling largescale robot teams to arrange themselves in geometric formations. Robustness, scalability, and adaptability are inherent properties of the resulting complex behavior.

In [7], a set of potential functions are described for enabling a group of mobile robots to arrange themselves in a formation while navigating to a goal in an obstacle field. According to the shape of the formation, every robot has several local attachment sites where other robots can be attached. The behavior of a robot is controlled by a group of motor schemes: move to goal, avoid static obstacles, avoid robots, and maintain formation. These motor schemes generate movement vectors, which are summed for computing the overall movement direction. The position of the robot is determined by building a list of all potential attachment sites; then, an attractive vector is generated towards the closest site. In this approach, new robots can be only integrated in free attachment sites; this thesis proposes an algorithm where robots are integrated wherever they meet a robot belonging to the formation.

A framework where local traffic rules are encoded through artificial potentials is presented in [22]. These artificial potentials define interactions forces between neighbors. The interaction forces maintain the inter-vehicle spacing. For controlling the movement of vehicles into the formation, virtual leaders are defined using local potential fields. There is no leader among the vehicles. The position of the vehicles is not defined a priori. Schooling and flocking maneuvers are presented for group translation and motion respectively. Switching between formations and avoiding obstacles are not treated in this work.

Krishnanand and Ghose [20] present an approach where the term of "local templates" is introduced. A local template is defined for every formation pattern: grid, line, and wedge. This local template is maintained by every robot. The templates encode information into multiple sectorial regions in order to generate virtual links between neighbors. These virtual links are detected and followed by other robots. The parameters that influence the shape of the formation include the distance between neighbors, and the angle between robot heading direction and the line-of-sight. The basic formation behaviors followed by each robot to lead to the desired global formation are: safe wandering, broadcast, sense neighbor, and align. New elements can be added only at free attachment points, and balance maneuver of formation is not allowed.

## 2.4 Discussion

After analyzing the several approaches proposed for solving the problem of robot formation, the problems considered in this thesis in order to contribute to the state of the art of the

robot formation control, include the possibility of that new elements can be added to the formation at anytime, everywhere [3], and the shape of the formation can be reconfigured; also, to remove the need of fixed positions for robots in formation in order to allow recovery after fails, and to include switching formation on the fly in order to robots can be adapted to the environment.

# Chapter 3

# An Algorithm for Agent Formation Control

## 3.1 Introduction

In this chapter, an algorithm for establishing a formation of mobile agents is presented. The mobile agents are distributed randomly in an obstacle-free workspace. The agents have knowledge neither of number of agents in formation nor the position to follow into formation. Every agent can communicate with its neighbors; the actions performed by an agent affects directly to its neighbors. Considering the role and position of their neighbors, every agent decides its role and its position. Four roles can be played by each agent: free, leader, left-follower, and right-follower. The leader role is always performed by the agent located at the middle of the formation, except in a column formation where the leader is at front of the column; the role played by each agent is known only by its neighbors. Three formations are considered: line, wedge, and column (Figure 3.1). Desired formation is obtained by the interaction between agents whose behavior is governed by a set of rules to change their state. Since position of agents is not fixed and not depends on the identifier, new agents can be added at any time.

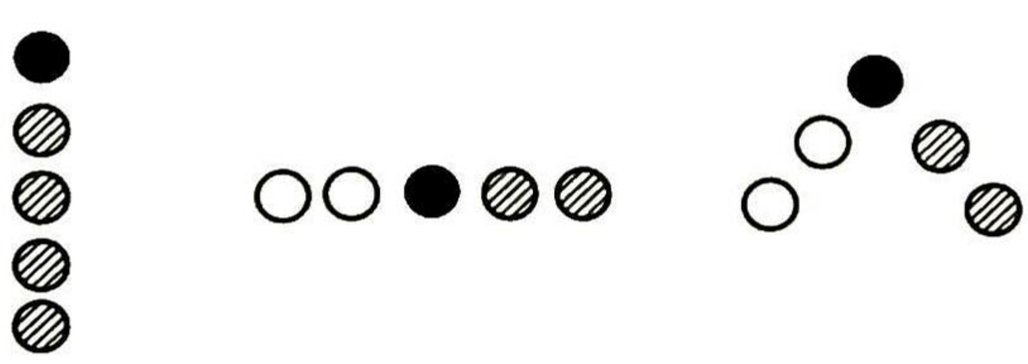


Figure 3.1: Black, dashed, and white agents are leaders, right followers, and left followers respectively.

The proposed algorithm is further described in next sections.

## 3.2 Operation

First problem encountered in the proposed scenario is the formation of a team of mobile agents in a meeting point. Since there is no knowledge about the number of agents in the scenario, the formation is configured according new elements are integrated. In order to allow the mobility of agents in the formation, two velocities are defined, *searching_velocity* and *formation_velocity*. The searching_velocity is adopted by free agents. This velocity is higher than the formation_velocity. The formation_velocity is adopted by agents in the formation. This velocity allows new elements to get their position into the formation while the agents are in movement.

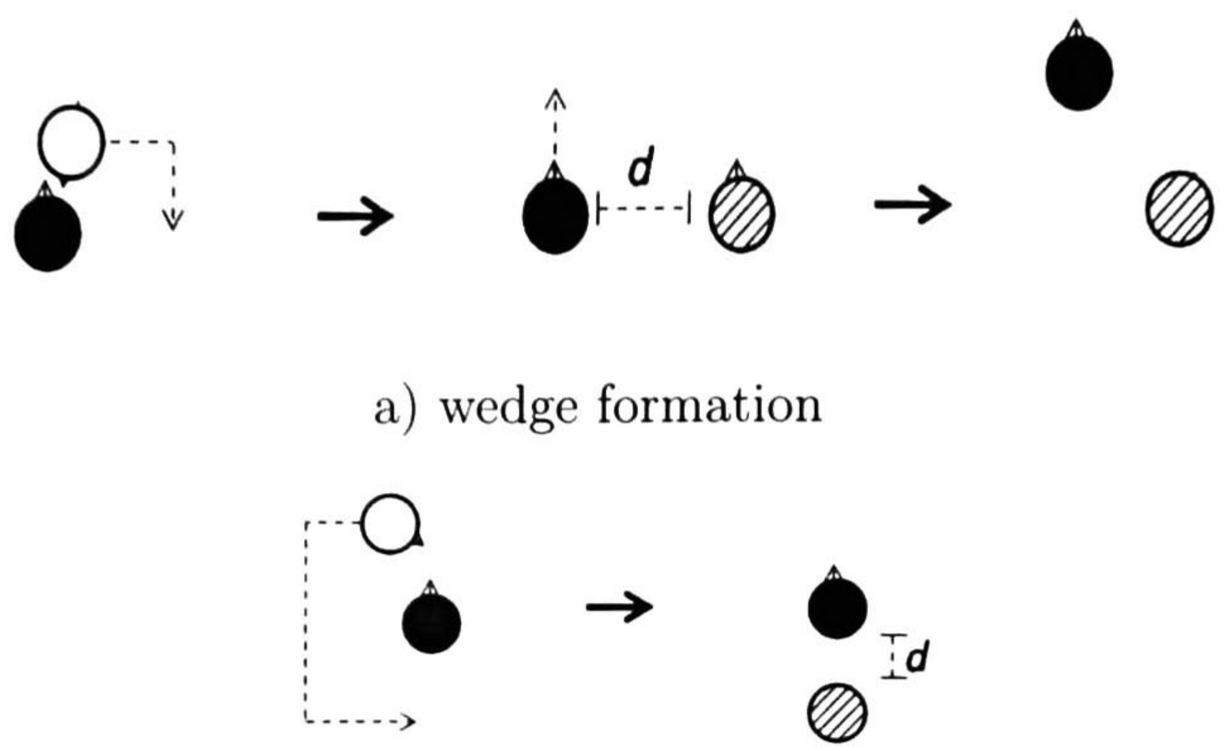
To initiate the formation, an agent is initialized with the information about the shape of the formation and the direction to follow. This agent emits a signal, which can be detected by other agents, while it moves to a goal. Every agent can perform one of the following roles:

- *Free.* An agent with this role maintains a searching_velocity while explores the environment looking for agents in a formation.
- *Leader.* The agent, which plays this role, identifies whether a balance maneuver is required and it decides if a switching formation has to be accomplished.
- *Right_Follower.* This role is played by an agent which is located to the right with respect to the leader.
- *Left_Follower.* This role is played by an agent which is located to the left with respect to the leader.

It is assumed that the signal emitted by the agents in the formation can be detected by all the agents who are trying to establish a formation. Once the signal is detected, the agents tend to move towards where signal strength is higher. When a free agent finds an agent in the formation, it starts a negotiation process in order to define its position into the formation. The position is defined according to the role of the agent, which has been discovered, in the formation as follows:

**The agent in the formation is a leader without neighbors.** In this case, the free agent moves to the right of the leader in wedge and line formations, or behind of the leader in a column formation. Once the agent has taken its position into the formation, it updates its heading to the same orientation of the heading of the leader (Figure 3.2). A distance $d$ is maintained with respect to other agents in order to avoid collisions between agents in the

formation and those which are being integrated. In a wedge formation, once the new agent has taken its position, the leader goes forward $d$ units.




a) wedge formation

b) column formation

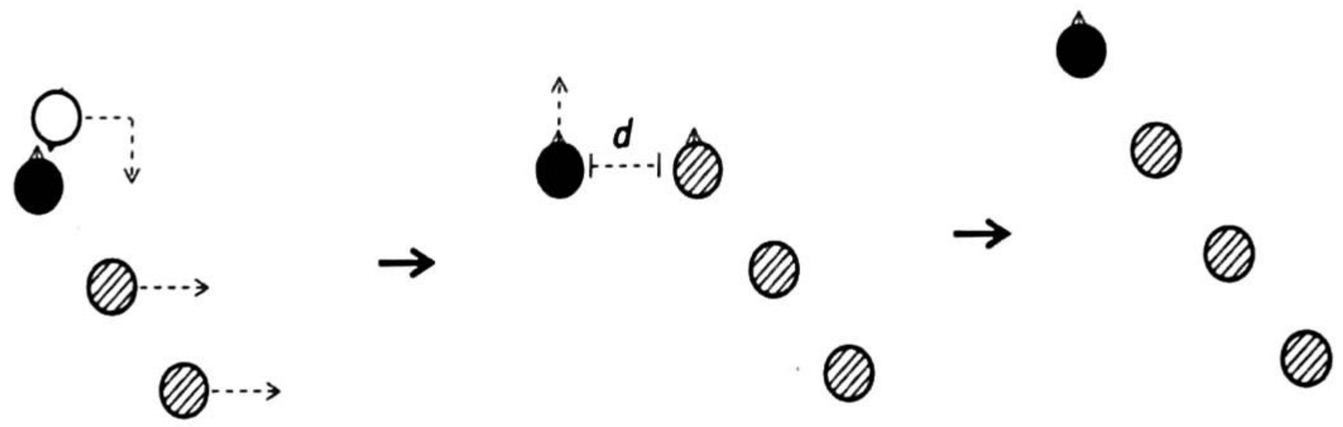
Figure 3.2: A free agent getting its position in a formation.

**The agent in the formation is a leader with a right_follower as a neighbor.** In this case, before the free agent moves to the right of the leader in wedge and line formations, or behind of the leader in a column formation, the agents, which are playing a right_follower role, are shifted to the right or rearward according to the formation shape. On the other hand, the leader and the left_followers will go forward $d$ units. (Figure 3.3).

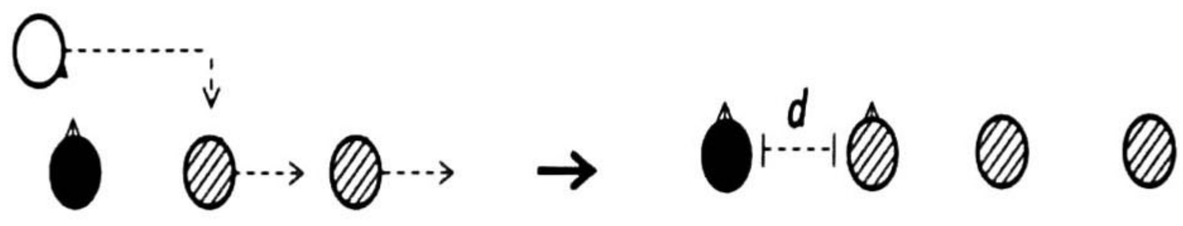
**The agent in the formation is the last right_follower in the formation.** When a free agent finds to another agent in this situation, the free agent moves to the right of the follower in wedge and line formations, or behind of the follower in a column formation. Agents, which are located to the left of the agent which is negotiating, will go forward one position ($d$ units) in a wedge formation (Figure 3.4).

**The agent in the formation is a right_follower but it is not the last.** When a free agent finds to another agent in this situation, before the free agent moves to the right of the follower in wedge and line formations, or behind of the follower in a column formation, the agents, which are playing a right_follower role and are located to the right of the agent which is negotiating, are shifted to the right or rearward according to the shape of formation. Agents, which are located to the left of the agent which is negotiating, will go forward one position ($d$ units) in a wedge formation (Figure 3.5).

**The agent in the formation is the last left_follower in the formation.** In this case, the free agent moves to the left of the follower in wedge and line formations, or behind of the follower in a column formation. Agents, which are located to the right of the agent

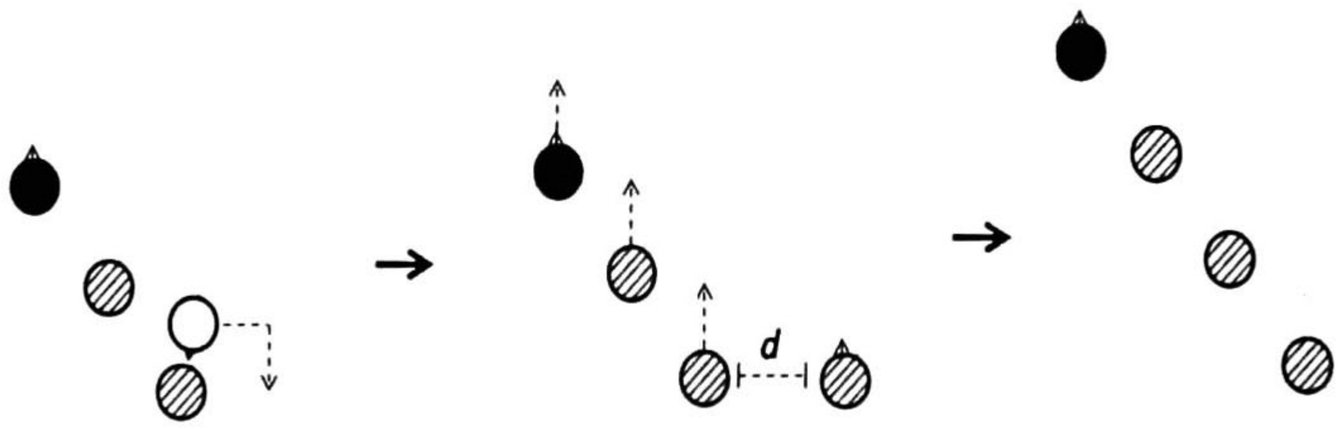



a) wedge formation

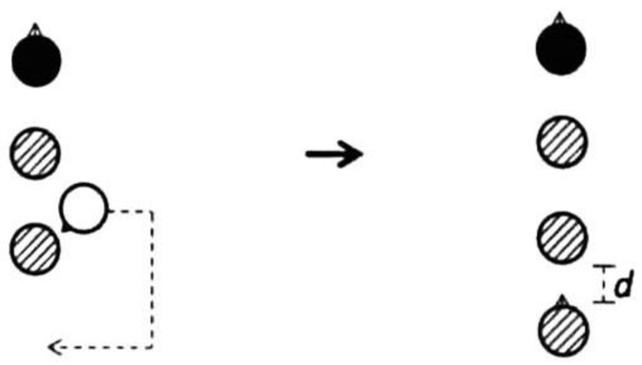



b) line formation

Figure 3.3: A free agent getting its position when the leader has a right_follower as a neighbor.

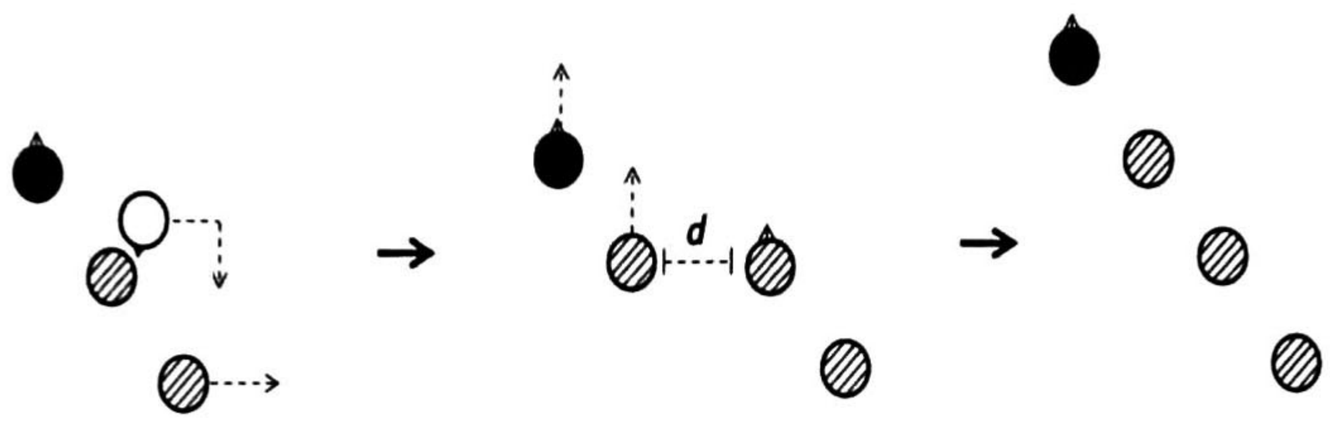



a) wedge formation

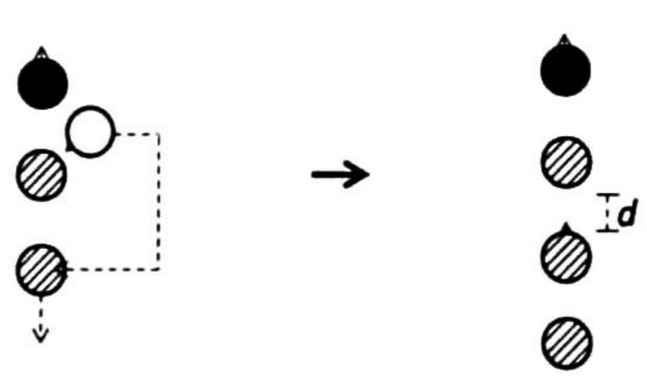



b) column formation

Figure 3.4: A free agent getting its position through negotiations with a right_follower.



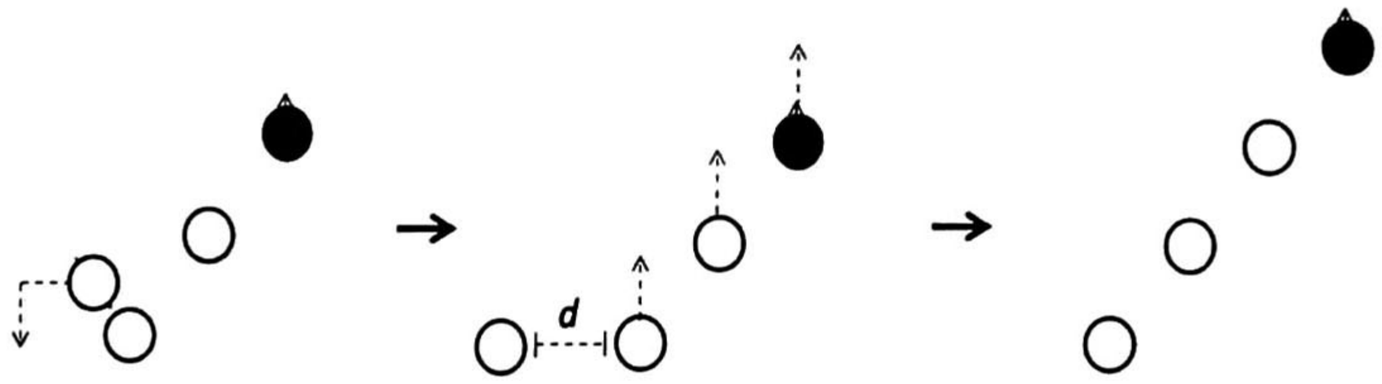

a) wedge formation




b) column formation

Figure 3.5: A free agent getting its position through negotiations with a right_follower which is not the last in the formation.

which is negotiating, will go forward one position in a wedge formation (Figure 3.6).




a) wedge formation

b) line formation

Figure 3.6: A free agent getting its position through negotiations with a left_follower.

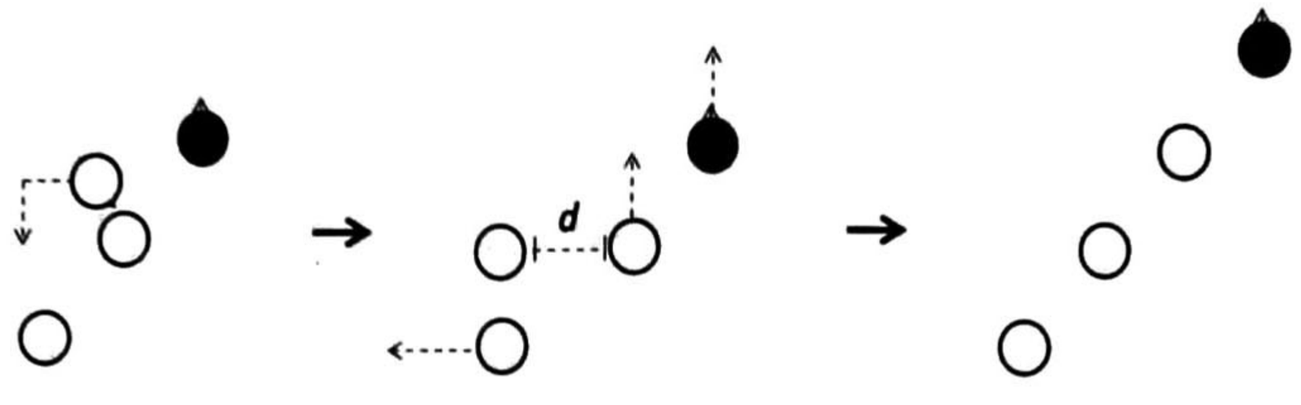
**The agent in the formation is a left_follower but it is not the last.** In this case, before the free agent moves to the left of the follower in wedge and line formations, or behind of the follower in a column formation, the agents, which are playing a left_follower role and are located to the left of the agent which is negotiating, are shifted to the left or rearward according to the shape of formation. Agents, which are located to the right of the agent which is negotiating, will go forward one position in a wedge formation. (Figure 3.7).

Once the agent has reached its correct pleace and updated its heading, it changes its search_velocity to a formation_velocity in order to move at the same velocity of the rest of the group.

### 3.2.1 Formation Balancing

New agents are integrated to the formation at the place where they meet another agent in the formation. This behavior produces a shape deformation. Therefore, the agent playing the leader role in a line formation has to be updated; on the other hand, a balance maneuver has to be accomplished in a wedge formation. The column formation does not suffer changes due that new agents are integrated behind of the leader.

Every time a new agent is integrated in a line formation, a message is forwarded until the leader receives it; once the leader has realized that a new agent has been integrated, it updates



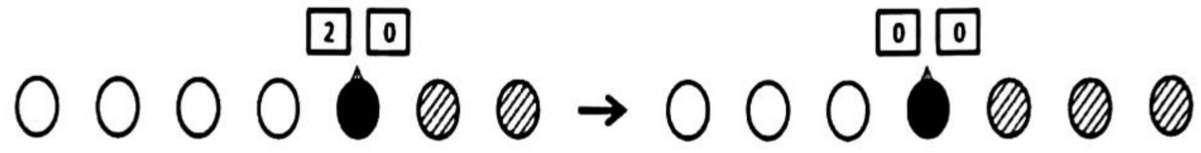

a) wedge formation

b) line formation

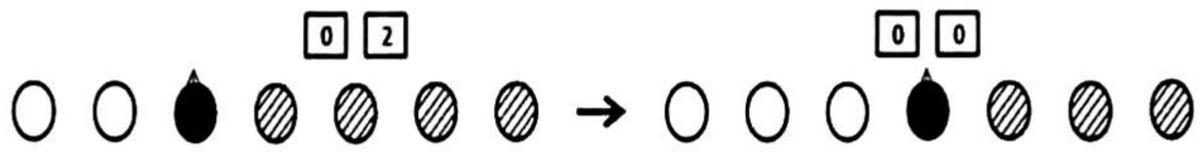
Figure 3.7: A free agent getting its position through negotiations with a left_follower which is not the last in the formation.

its counter of left and right neighbors; if both counters are equals, they are reinitialized to zero. If one of the counters has a difference of two with respect to the another, then a process for updating the leader role is started. This process consists on to change the role of the leader and ask to the corresponding neighbor be the new leader. If the counter of left neighbors is greater than the counter of right neighbors, then the leader changes its role to right_follower and its left neighbor changes its role to a leader. On the other hand, if the counter of right neighbors is greater than the counter of left neighbors, then the leader changes its role to left_follower and its right neighbor changes its role to a leader. Figure 3.8 illustrates this process.

For balancing a wedge formation, the same process is performed with some additions. Instead of changing only the role of agents, a displacement is accomplished in order to reconfigure the shape of the formation. If the counter of left neighbors is greater than the counter of right neighbors, before updating the roles as mentioned before, all the agents playing a left_follower role will go forward two positions (one position is equivalent to go forward a distance $d$). On the other hand, if the counter of right neighbors is greater than the counter of left neighbors, then all the agents playing a right_follower role will go forward two positions. Figure 3.9 illustrates this process.

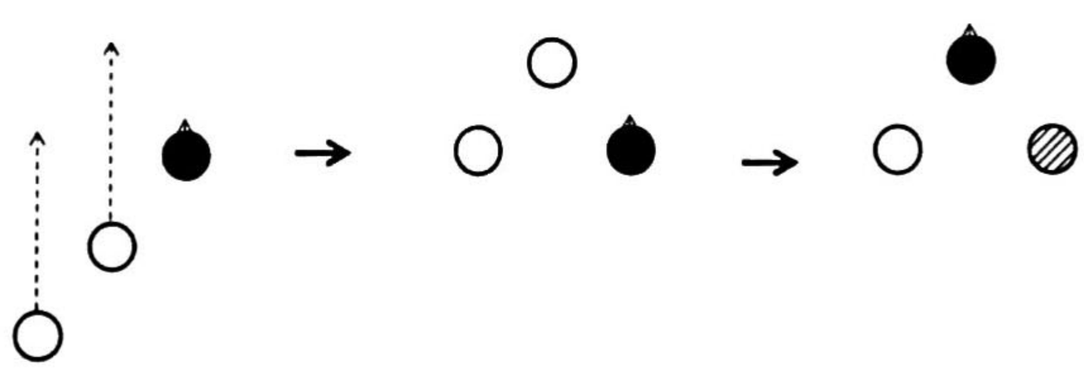



a) left unbalanced

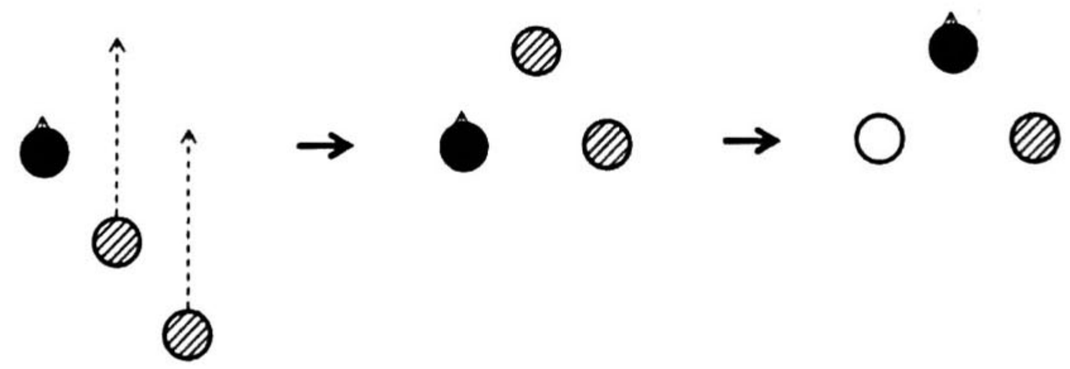



b) right unbalanced

Figure 3.8: Process for updating the leader role.



a) left unbalanced



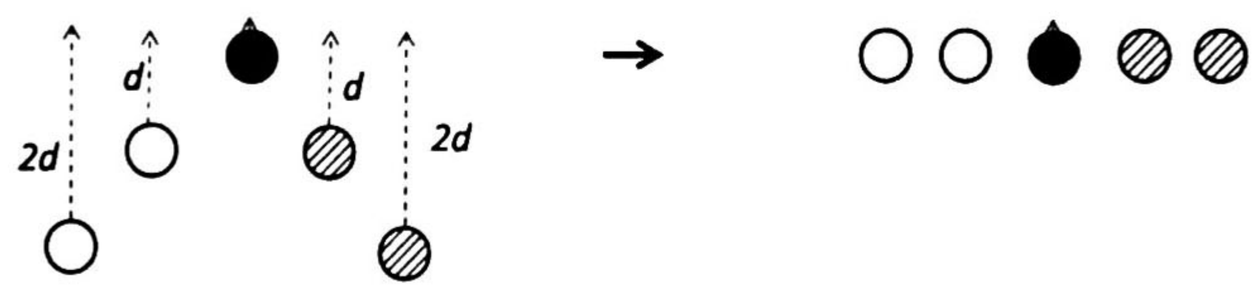
b) right unbalanced

Figure 3.9: Process for updating the leader role and balancing a wedge formation.

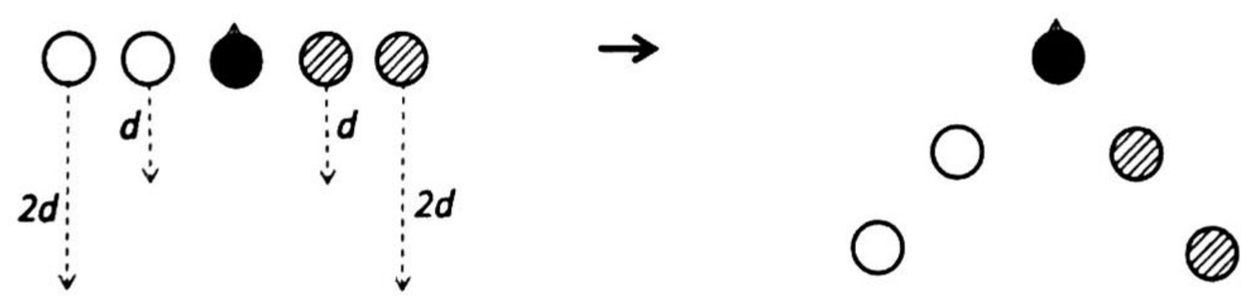
### 3.2.2 Switching between Formations

During navigation in hazardous and unknown environments, the possibility of adapting the formation to the environment by switching between formations brings as consequence a robust and adaptable behavior group; this behavior allows the completion of the task assigned to the agents. In this thesis the possibility of changing between line, wedge, and column formation is presented. When an instruction, which indicates the formation to perform, is received, the switching process is initiated by the leader; the autonomous decision of changing the formation according to the environment is not included; however, it is considered as future work.

In order to switch from a wedge formation to a line formation, the leader starts the process by sending a message to its neighbors for going forward a distance $d$. The neighbors, which receive this messages, increase the distance by $d$ units and send the message again to their left or right neighbors according to the role of the agent which sends the message. An agent, which is playing a right_follower role, sends the message to its right neighbor, while an agent, which is playing a left_follower role, sends the message to its left neighbor. A representation of this process is shown in Figure 3.10a. For switching from a line formation to a wedge formation, the process described above is performed, but instead of going forward, the agents stop during the time needed for advance the assigned distance according to the formation velocity (Figure 3.10b).




a) wedge to line

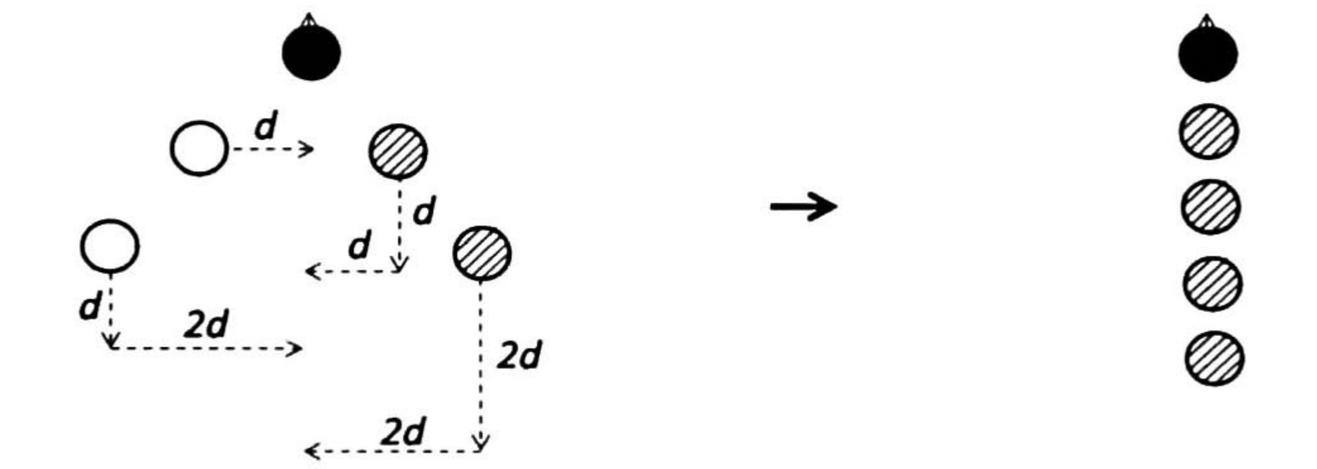



b) line to wedge

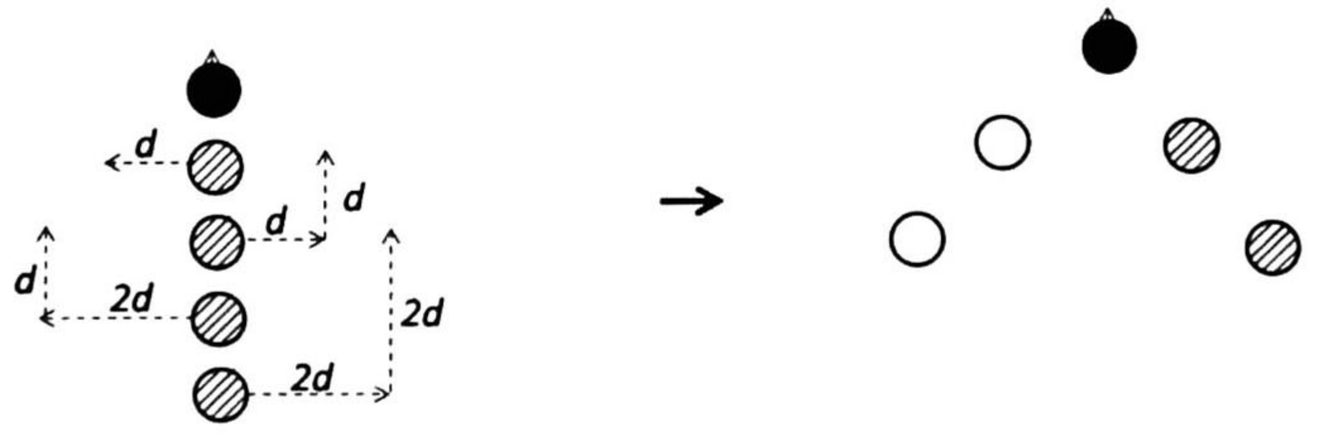
Figure 3.10: Process for switching between wedge and line formations.

To switch from a wedge formation to a column formation two distances are transmited, one for stopping and the other for moving to the left or to the right. The leader sends a message to its right neighbor with both distances initialized to $d$; the message sent to the left neighbor initializes the distance for stopping to 0 and the distance for moving to $d$ units. The agents, which receive this message, increase the distance by $d$ units. Agents which are playing a right_follower role move to the left, while agents which are playing a left_follower role move to the right (Figure 3.11a).

For switching from a column formation to a wedge formation two distances are transmitted, one for going forward and the another for moving to the left or to the right. Both distances are initialized to 0. A counter is added to the message in order to determine which agents move to left and which agents move to right. The counter is increased every time it is forwarded. If the module two of the counter received by the agent is 0, then the distance for going forward is increased by $d$ units and the agent moves to the right. On the other hand, if the module two of the counter received is different to 0, then the distance for moving to a side is increased by $d$ units and the agent moves to the left (Figure 3.11b).




a) wedge to column




b) column to wedge

Figure 3.11: Process for switching between wedge and column formations.

In order to switch from a line formation to a column formation two distance values are

transmitted. The first indicates how long time the agent has to stop its movement. Second value indicates the distance to be travelled by the agent. The sent values are initialized by the leader. Both values are initialized to $d$ for the left followers; for right followers the stop value is initialized to $2d$ and the movement value is initialized to $d$. The stop value is increased by $d$ units for every agent, and the movement value is increased by $2d$ units. Right followers move to the left and left followers move to the right. This process is illustrated in Figure 3.12a.

To switch from a column formation to a line formation two distance values are transmited. The first indicates the distance to be travelled to the leader direction. Second value indicates the distance to be travelled to the left or to the right. Both values are initialized by the leader to $d$. A counter is added to the message in order to determine which agents move to left and which agents move to right. The counter is increased every time it is forwarded. If the module two of the counter received by the agent is 0, then the distance for going forward is increased by $d$ units, and the distance for move to a side is not increased; in this case, the agent moves to the right. On the other hand, if the module two of the counter received is different to 0, then the distance for moving to a side, and the distance for going forward are increased by $d$ units; in this case, the agent moves to the left (Figure 3.12b).

## 3.3 Algorithm Formalization

In the previous section, an informal description of the proposed algorithm was presented. In this section a formal description is presented using Petri Nets. A model for the agent controller is proposed. This model describes the variable states, which are synchronized between them, for representing the global status of every agent. Furthermore, how simple rules can be obtained from this model is described.

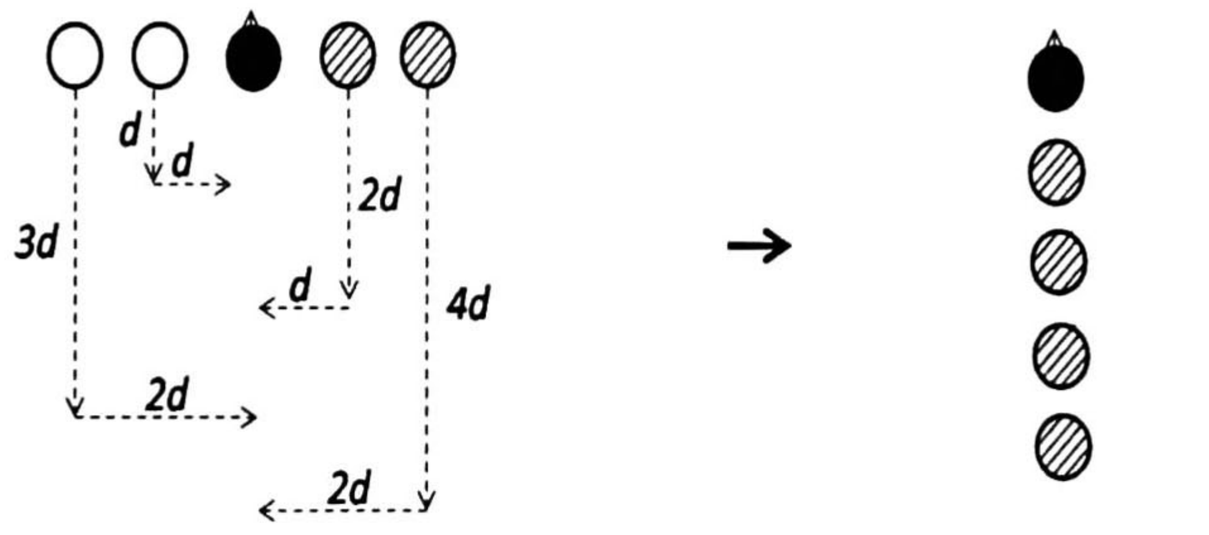
### 3.3.1 Mobile Agent State

For representing the state of the agent and external signals, six state variables are defined: formation, role, status_of_formation, alignment, role_of_neighbor, and balance.

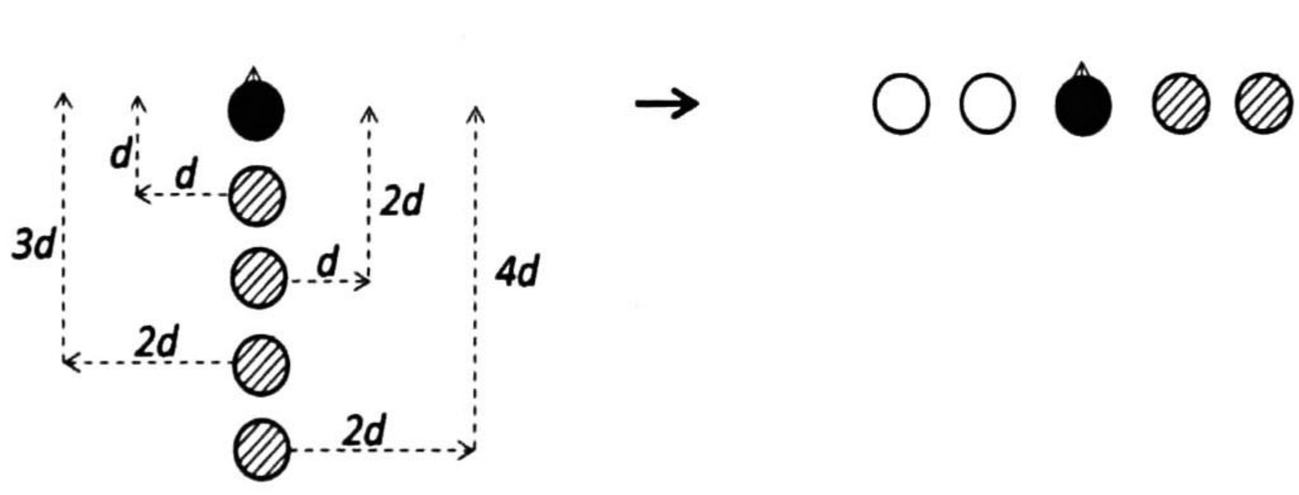
**Formation (F).** This variable represents three possible states which represent the formation being performed: line, wedge, or column. When a switching formation process is performed the value of this variable is updated.

**Role (R).** This variable maintains the information about the role being played by an agent, leader or follower. When an agent gets into formation, it updates its role to follower. During a balance maneuver, agent role can be updated.

**Status_of_formation (SF).** This variable indicates if the agent has been included in a




a) line to column




b) column to line

Figure 3.12: Process for switching between line and column formations.

formation or whether is still free. Agent in a free status looks for agents which have part of a formation. When two free agents met, they consider other agent as an obstacle and try to avoiding it in order to look for an agent in formation.

**Alignment (A).** A follower agent is identified as right or left follower according to its position within the formation. When a balance maneuver is performed, if the agent is a leader or a left follower with a neighbor as leader, then this variable is updated.

**Role_of_neighbor (RN).** This variable represents the role of the neighbor located ahead the agent; i.e. for an agent with a left alignment this information corresponds to right neighbor. This information is used to identify the role to be taken when a balance maneuver is performed.

**Balance (B).** This variable is modified only by a leader agent; The leader stores the information needed to know when a formation is unbalanced and for determining which side has more elements. The status of this variable is changed every time new agents are added into the formation.

### 3.3.2 Agent Controller

Every agent has an integrated controller, which gives the possibility of reacting to the events and signals detected by the agent from the environment or from other agents. This controller is the same for all the agents which are participating in the formation process. The interactions between agents give as a result the complex behavior induced by the state variables of the agents.The agent controller, which is presented here, is defined as follows:

SYSTEM_COMPONENTS = {agent_controller}. The state variables described above (F, R, SF, A, RN, and B) will describe the global state of the agent. The sets {*wedge, line, column*}, {*leader, follower*}, {*free, in_formation,*}, {*right, left*}, {*leader, follower*}, and {*balanced, semi_balanced, right_unbalanced, left_unbalanced*} are the values that the state variables F, R, SF, A, RN, and B can take, respectively. In order to obtain the initial marking, it is assumed that the initial state for the system is F = *wedge*, R = *leader*, SF = *free*, A = *right*, RN = *leader*, and B = *balanced*. For the agent, which is initialized with the information about the shape of the formation and the direction to follow, the variable SF is initialized with the *in_formation* value, and the variable F is initialized according to the shape of formation to establish. The PN modules obtained are depicted in Figure 3.13.

The set of rules is straightforward derived from the PN model obtained from the synchronous composition of the PN models representing every state variable. Transitions labels declare that those transitions having the same symbolic label must be synchronized, i.e, merged; transitions that have more than one label must be replicated for synchronizing. For the sake of readability, the implicit composition is expressed by the labeling. Each rule is

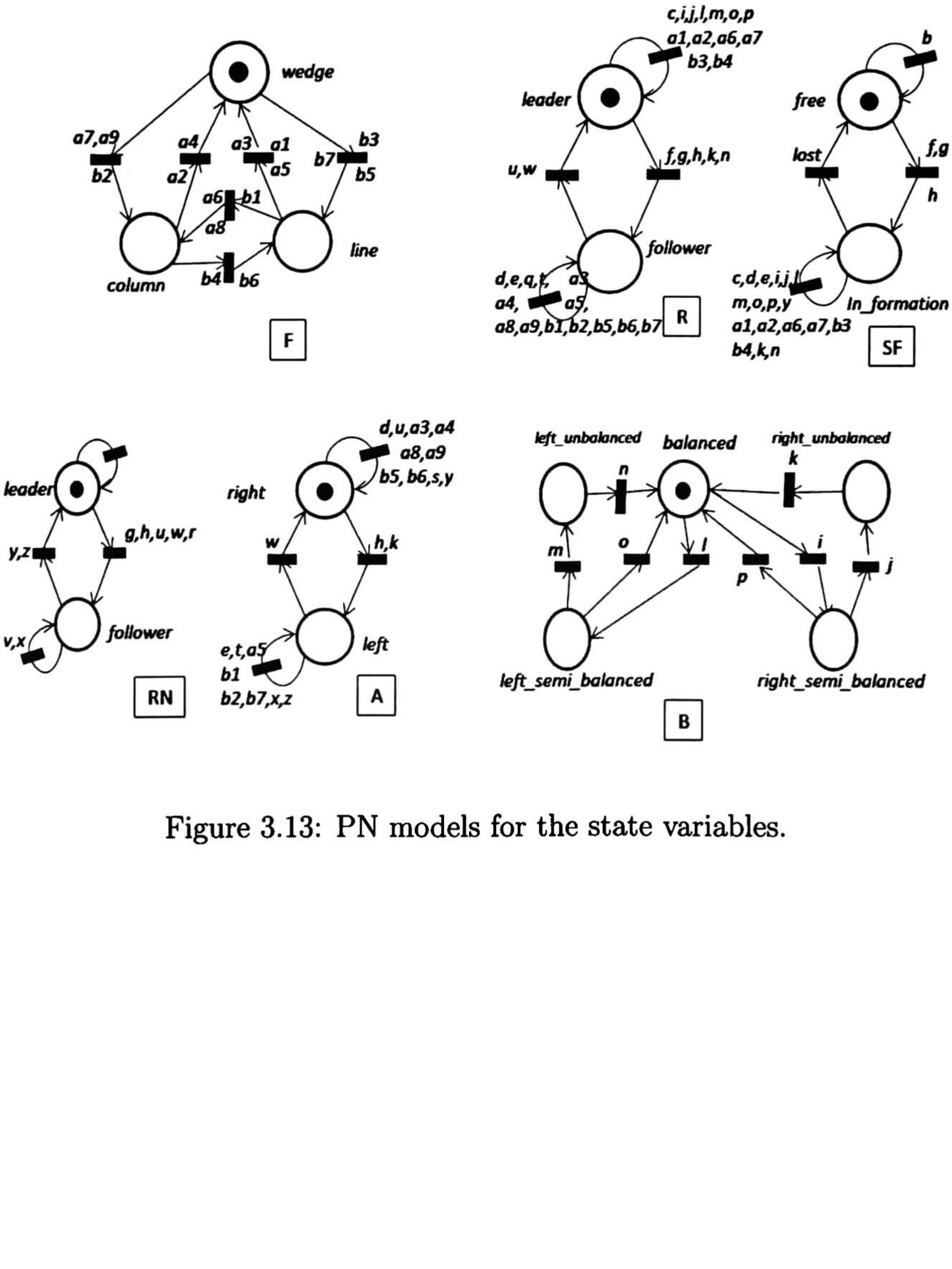



Figure 3.13: PN models for the state variables.

then built from each transition; the condition part includes a combination of state variables values, given by the enabling marking, and the detection of a pertinent input signal; the action part consist of the generation of an output signal to the agent neighbors, the execution of an action (optionally), and the updating (given by the firing transition mechanism) of the involved state variables. The actions which can be performed by the agents are described in Table 3.1. For the stated problem Table 3.2 includes the signals associated to labels of the PN model for integrate a new agent into a formation. Table 3.3 includes the signals associated to labels of the PN model for balancing and updating the leader role. Table 3.4 includes the signals for switching between formations.

B, C, and D-agents represent signals emitted by an agent with a global status described as follows: B-agent, signal emitted by an agent defined as leader; C-agent, signal emitted by an agent defined as right follower; D-agent, signal emitted by an agent defined as left follower.

In order to maintain communication with its neighbors, every agent stores the address of its left and right neighbor in the LeftN and RightN variables, respectively. The direction to follow is stored in the Heading variable.

From the PN model presented above, the labelling synchronization, and the input signals, behavior rules can be generated. These rules allow to the agent for deciding the actions to perform. This interaction between agents brings as a result the desired global behavior(formation shape assigned) in the group of mobile agents. As an example, three rules obtained following the previous procedure are included below.

```
If ( Answer C-Agent && SF = free )
{
        R = follower
        SF = in_formation
        RN = follower
        LeftN = messageAddress
        RightN = RightN_Received
        Move(Right)
        // Send parameters: Signal, dest_address,left_neighbor,
        // right_neighbor, formation_direction
        Send(L-Forward, LeftN, LeftN, RightN, Heading )
        Send(R-Move, RightN, LeftN, RightN, Heading)
}
```

Table 3.1: Actions which can be performed by the agents.

| Action | Description |
|---|---|
| A++ | Increase the distance of A in *d* units |
| A+2 | Increase the distance of A in *2d* units |
| B+ | Increase the distance of B in *d* units if C mod 2 != 0 |
| B++ | Increase the distance of B in *d* units |
| C++ | Increase the counter in 1 unit |
| B | Broadcast the output signal |
| SB | Send the output signal to both neighbors |
| SA | Send the output signal to the address of the input signal |
| SL | Send the output signal to the left neighbor |
| SR | Send the output signal to the right neighbor |
| MCB | Move until cover the distance received in B, to the *right* if C mod 2 = or to the *left* in other case |
| ML | Move to the left |
| MLB | Move to the left until cover the distance received in B |
| MR | Move to the right |
| MRB | Move to the right until cover the distance received in B |
| F | Go forward one position |
| F2 | Go forward two positions |
| FA | Go forward until cover the distance received in A |
| WA | Wait the time corresponding to go forward until cover the distance A |
| LNLN | Store the address received in the *ln* variable as left neighbor |
| LNSA | Store the address of the input signal as left neighbor |
| RNRN | Store the address received in the *rn* variable as right neighbor |
| RNSA | Store the address of the input signal as right neighbor |
| SBFD | Send to both neighbors the distance to go forward initialized to *d* |
| SRC | Send to the right neighbor a counter initialized to C |
| SRC1 | Send to the right neighbor a counter initialized to 1 |
| SRFA | Send to the right neighbor the distance to go forward initialized to A |
| SRFD | Send to the right neighbor the distance to go forward initialized to *d* |
| SRMB | Send to the right neighbor the distance to move initialized to B |
| SRMD | Send to the right neighbor the distance to move initialized to *d* |
| SRW0 | Send to the right neighbor the distance to wait initialized to 0 |
| SRW2D | Send to the right neighbor the distance to wait initialized to *2d* |
| SRWA | Send to the right neighbor the distance to wait initialized to A |
| SRWD | Send to the right neighbor the distance to wait initialized to *d* |
| SLFA | Send to the left neighbor the distance to go forward initialized to A |
| SLMB | Send to the left neighbor the distance to move initialized to B |
| SLMD | Send to the left neighbor the distance to move initialized to *d* |
| SLW0 | Send to the left neighbor the distance to wait initialized to 0 |
| SLWA | Send to the left neighbor the distance to wait initialized to A |
| SLWD | Send to the left neighbor the distance to wait initialized to *d* |

Table 3.2: Conditions for firing synchronized transitions of Petri Net, the signals produced, and the actions to be performed in order to integrate a new agent.

| Input signal | Firing label | Output signal | Actions |
|---|---|---|---|
| Discovered agent | b | Request information | B |
| Request information | c | Answer B-agent | SA |
| Request information | d | Answer C-agent | SA |
| Request information | e | Answer D-agent | SA |
| Answer B-agent | f | L-forward, R-move | LNSA, RNRN, MR, SL, SR |
| Answer C-agent | g | L-forward, R-move | LNSA, RNRN, MR, SL, SR |
| Answer D-agent | h | R-forward, L-move | RNSA, LNLN, ML, SR, SL |
| L-forward | i,j,o | L-forward | F, RNSA, SL |
| R-forward | l,m ,p | R-forward | F, LNSA, SR |
| L-forward | q | L-forward | F, RNSA, SL |
| R-forward | q | R-forward | F, LNSA, SR |
| R-move | s | R-move | MR, LNSA, SR |
| L-move | t | L-move | ML, RNSA, SL |

Table 3.3: Conditions for firing synchronized transitions of Petri Net, the signals produced, and the actions to be performed in order to balance the formation.

| Input signal | Firing label | Output signal | Actions |
|---|---|---|---|
| None | k | R-Balance | SR |
| None | n | L-Balance | SL |
| R-Balance | v | R-Balance | F2, SR |
| R-Balance | u | R-Balance, LR-Update | F2, SR, SB |
| L-Balance | x | L-Balance | F2, SL |
| L-Balance | w | L-Balance, LR-Update | F2, SL, SB |
| LR-Update | y | Update | SR |
| LR-Update | z | Update | SL |
| Update | r | None | None |

Table 3.4: Conditions for firing synchronized transitions of Petri Net, the signals produced, and the actions to be performed in order to switch between formations.

| Input signal | Firing label | Output signal | Actions |
|---|---|---|---|
| W-Switch | a1 | W-Switch | SB, SBFD |
| W-Switch | a2 | W-Switch | SB, SRW0, SRMD, SRC1 |
| W-Switch | a3 | W-Switch | SR, WA, A++, SRWA |
| W-Switch | a4 | W-Switch | SR, WA, MCB, A++, B+, C++, SRWA, SRMB, SRC |
| W-Switch | a5 | W-Switch | SL, WA, A++, SRWA |
| C-Switch | a6 | C-Switch | SB, SLWD, SLMD, SRW2D, SRMD |
| C-Switch | a7 | C-Switch | SB, SLW0, SLMD, SRWD, SRMD |
| C-Switch | a8 | C-Switch | SR, WA, MLB, A+2, B++, SRWA, SRMB |
| C-Switch | a9 | C-Switch | SR, WA, MLB, A++, B++, SRWA, SRMB |
| C-Switch | b1 | C-Switch | SL, WA, MRB, A+2, B++, SLWA, SLMB |
| C-Switch | b2 | C-Switch | SL, WA, MRB, A++, B++, SLWA, SLMB |
| L-Switch | b3 | L-Switch | SB, SBFD |
| L-Switch | b4 | L-Switch | SB, SRFD, SRMD, SRC1 |
| L-Switch | b5 | L-Switch | SR, FA, A++, SRFA |
| L-Switch | b6 | L-Switch | SR, FA, MCB, A++, B+ C++, SRFA, SRMB, SRC |
| L-Switch | b7 | L-Switch | SL, FA, A++, SLFA |

If ( *R-Balance* && RN = *follower* )

{

Forward(2d)

// Send parameters: Signal, dest_address,left_neighbor,

// right_neighbor, formation_direction

Send(*R-Balance*, RightN, LeftN, RightN, Heading )

}

If ( *R-Balance* && R = *follower* && A = *right* && SF = *in_formation* && B = *balanced* && F = *wedge*)

{

F = *line*

Forward(A)

A++

SendDistance(RightN, A)

// Send parameters: Signal, dest_address,left_neighbor,

// right_neighbor, formation_direction

Send(*L-Switch*, RightN, LeftN, RightN, Heading )

}

Agents can detect all the signals produced by their neighbors; these signals can be represented in a binary format requiring four bits for representing all the signals; thus, they are transmitted in a message or in another signaling method that could be decoded by agents. Furthermore, every agent maintains information about the orientation to follow and the address to communicate with its left and right neighbors. this information is transmitted and updated every time new agents are integrated into formation. Orientation of the formation is defined by the agent defined as the initial leader, and this is maintained by all new agents, which receive it from their neighbors.

## 3.4 Obstacle Avoidance

Although switching between formations gives a desirable adaptability to the formation of agents, it is unavoidable to find a diversity of obstacles in the environment, which could decrease the performance of the agent when they are accomplishing an assigned task. Several works have been proposed for solving the problem of avoiding obstacles when a mobile robot navigates towards a goal. Some works, which could be integrated to the agents which are performing the algorithm proposed here, are described in [27, 17, 12, 4]. Any of these algorithms can be applied because the controller only reacts to specific signals described above, then an obstacle avoidance algorithm can be performed in parallel.

However, while an agent is performing an obstacle avoidance algorithm, the contact with its neighbors is lost. If the obstacle is not avoided in a threshold time $t$, or if after evading the obstacle is not possible for the agent to recover its position into the formation, then the agent is reinitialized to a *free* state. In this state, agents again look for agents in a formation. In order to recover the position of the agent into the formation the following process is performed.

At the moment to start the obstacle avoidance process, the agent stores the formation_velocity $fv$ and the direction of the formation $\alpha$. A timer $ti$ is started. Every time the agent changes its direction with a deviation of $\pm 5$ degrees, the new direction $\beta$, and the velocity $v$ are stored, and a timer $c$ is initialized. When the direction is changed again, the timer is stopped and the distance $d$ is calculated:

$$d = \mathrm{v} * \mathrm{c}$$

then, the components of the vector obtained are computed,

$$v_x = \mathrm{d} * \cos\beta$$
$$v_y = \mathrm{d} * \sin\beta$$

and stored in a matrix

$$\mathrm{vectors[i][0]} = v_x$$
$$\mathrm{vectors[i][1]} = v_y$$

Once the obstacle has been avoided, the components of the vector, which indicates the distance and the direction that the agent should have followed whether the obstacle does not exist, are obtained

$$d = \mathrm{fv} * \mathrm{ti}$$
$$ov_x = \mathrm{d} * \cos\alpha$$
$$ov_y = \mathrm{d} * \sin\alpha$$

then, the sum of the stored vectors is computed

$$tv_x = \sum_{i=0}^{n} vectors[i][0]$$
$$tv_y = \sum_{i=0}^{n} vectors[i][1]$$

Finally, in order to recover the position of the agent, a vector of movement is obtained from the difference between the sum of the stored vectors ($tv$) and the original vector ($ov$).

$$mv_x = ov_x - tv_x$$
$$mv_y = ov_y - tv_y$$
$$mv = \sqrt{mv_x^2 + mv_y^2}$$
$$\theta = \arctan(mv_y/mv_x)$$

Once the agent moves the distance computed (mv) in the direction obtained, the obstacle avoidance process is finished.

# Chapter 4

# Formation Control Simulation

For validating the proposed algorithm, which was described in the previous chapter, two simulated scenarios have been implemented. First, a simulation was implemented in NetLogo [34] for evaluating the global behavior obtained from the interactions between agents. Second, a simulation was implemented in Webots [1] in order to evaluate the capability of mobile robots to perform a group formation.

Initally, a workspace is created and all the robots/agents are randomly distributed in the workspace. Every robot has a controller which is constructed based on the rules provided by the Petri net model and the tables described in chapter 3. Since the robots share no global coordinate system, their position in the formation is determined from the actual position of their neighbors.

Through these simulations, it can be observed how the desired formation is established due the interactions of agents. The simulation of agents does not consider the physical restrictions of movements, only behavior is considered. In the case of the simulated robots, a robot architecture is defined and a minimal set of sensors is specified in order to perform the actions which are required to get a position into the formation, to balance the formation, or to switch between formations. Both scenarios and the results obtained are described in the next sections.

## 4.1 NetLogo Simulation

NetLogo is a simulator for modeling and simulating natural and social phenomena. Global behaviors obtained by local interactions can be simulated through this simulator. In order to observe how the rules, which were described in previous sections, bring as consequence of interactions the desired formation of mobile agents, a simulation scenario is implemented.

In this simulation only the behavior, which is obtained from the rules, is evaluated; the actions needed for getting the position into the formation are assumed to be perfectly executed due the nature of the simulator. It can be observed how the agents in a wedge formation perform a balance maneuver every time new agents are integrated and the formation shape is unbalanced. It is possible to observe how the process of switching between formations is accomplished. The scenario used for this purpose is presented below.

### 4.1.1 Scenario Description

Simulation was performed with 11 agents establishing a wedge formation. Initially all agents are distributed randomly in the environment, and a initial point for starting the formation is represented by a black cell (Figure 4.1). When an agent detects the initial point, it establishes its role to leader. According new agents find a robot into formation, they are integrated to the group. The position of the new agent is determined by the actual state of its neighbor. If the agent in the formation is to the right, then the new agent will be joined to the right. On the other hand, if the agent in the formation is to the left, then the new agent will be joined to the left. Since agents can be integrated to the left or to the right of the formation, the shape of the formation could be unbalanced. An unbalanced formation is illustrated in Figure 4.1b.

A leaders verifies if formation is unbalanced every time new agents are added into formation. When an unbalanced formation is detected, a balance maneuver is performed. The role of the agents is updated after a balance maneuver is performed. A wedge formation after balance maneuver is shown in Figure 4.1c. After a while, all the agents get into formation. Balance maneuvers required during that process are performed in order to maintain the shape of formation. If new agents appear, they can be joined to the formation without problems. Final configuration of wedge formation is illustrated in Figure 4.1d.

At any time, the leader can decide to switch the actual formation to another shape allowed (wedge, column, line). A signal is emitted to its neighbors in order to cause a switching formation. A message indicating the distance to travel and/or the time to wait is sent. That information is forwarded to the neighbors of the agent. The results obtained frpom this behavior are illustrated in Figure 4.2. The behavior of switching from a line to a wedge is shown in Figure 4.3.

## 4.2 Webots Simulation

In order to evaluate the behavior obtained from the algorithm proposed here in mobile robots, a simulation was performed in Webots. This simulator allows to create robots with the shape

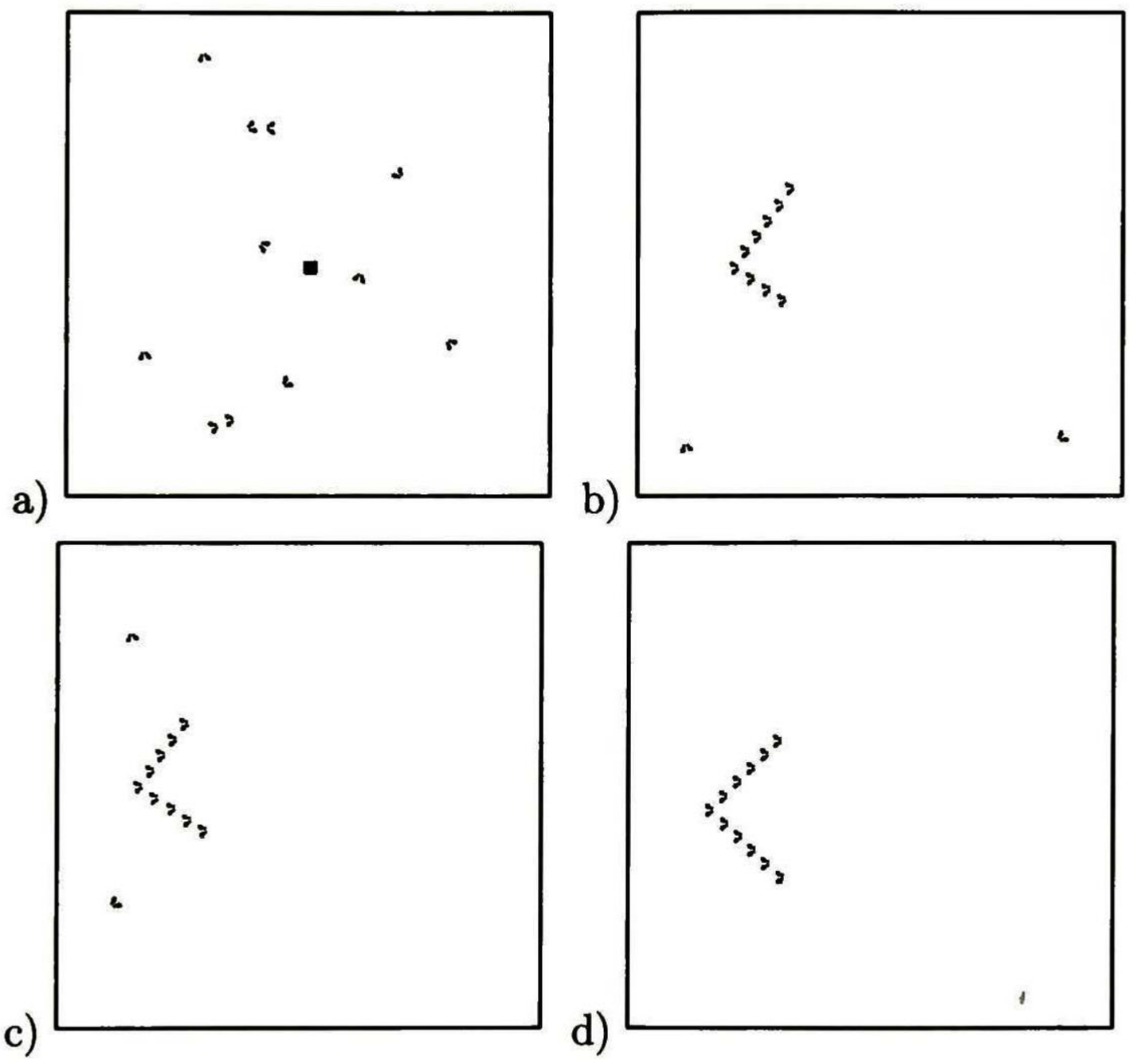



Figure 4.1: a) Agents distributed randomly in the environment. b) An unbalanced wedge formation. c) Agents in formation after balance maneuver. d) Wedge formation after all agents got their position.

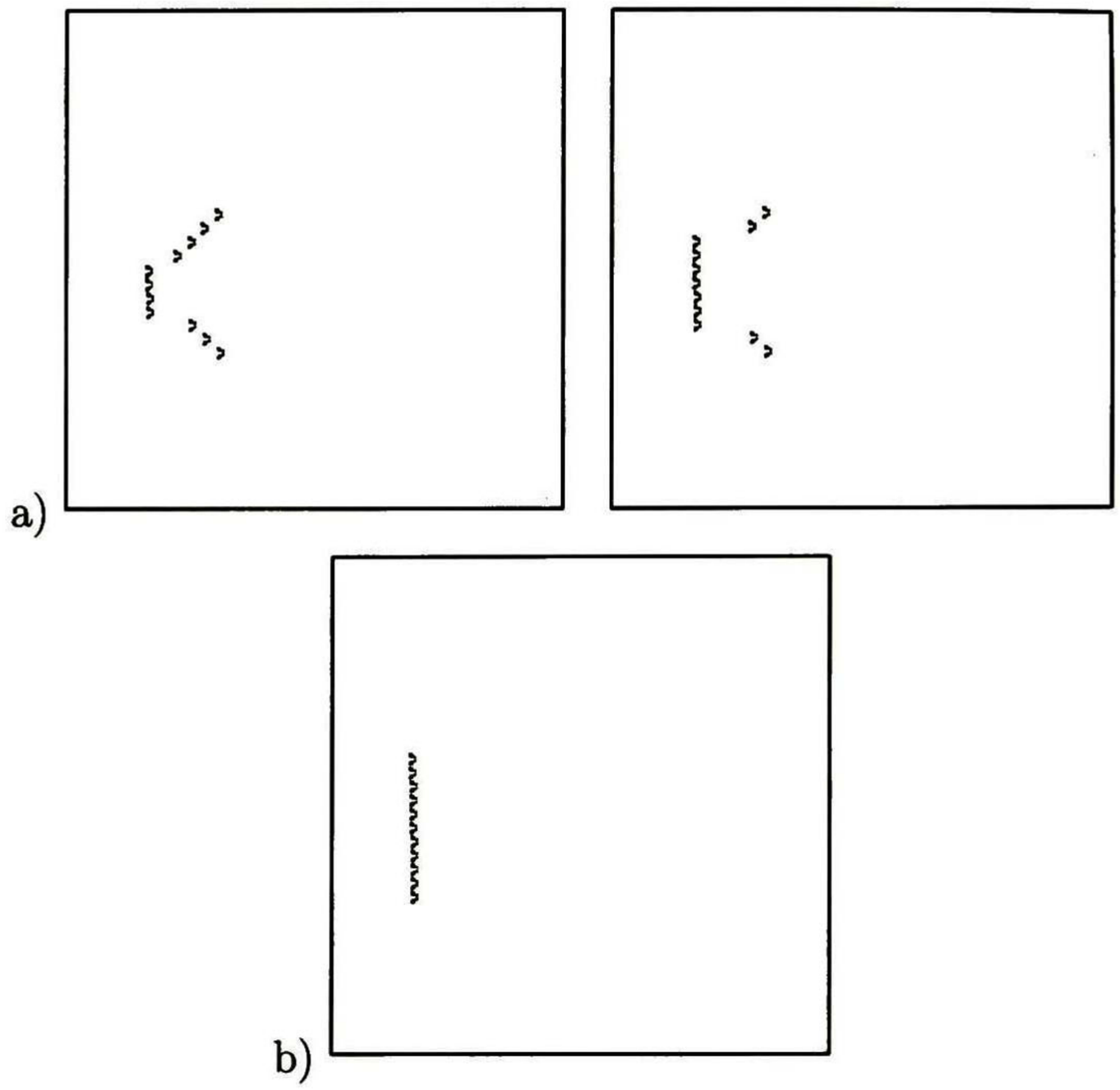



Figure 4.2: a) Switching from a wedge formation to a line. b) Agents into a line formation after the switching process.

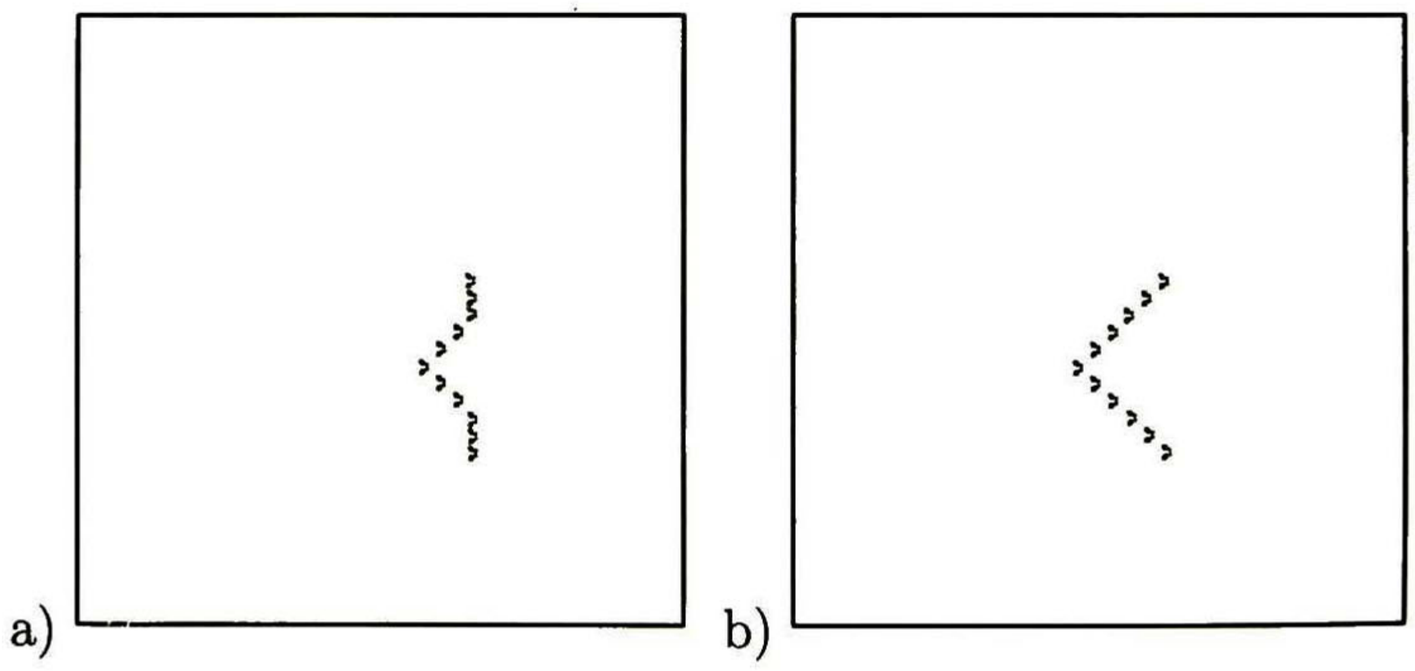



Figure 4.3: a) Switching from a line formation to a wedge. b) Agents into a wedge formation after the switching process.

and sensors needed for performing an especific task. With this simulator, it is possible to evaluate the behavior of robots with physical restrictions, where sensors have to be used in order to obtain information from the environment and react to it.

In this simulation, the actions needed in order to get a position into the formation are solved according to the sensors used in the model of robots here proposed. One of the advantages of the algorithm here proposed, is the flexibility to solve the physical problems for moving a robot to an assigned position according to the capabilities of the robot. While the rules of behavior are followed, the robots will be able to establish the assigned formation.

### 4.2.1 Robot architecture

Each robot has been designed to provide features of basic mobility, obstacle sensing, and local interactions. The number of sensors used in these robots are the minimal sensors needed in order to achieve the formation task with the algorithm proposed here. However, the algorithm presented only defines the actions to be followed by every robot in order to obtain the formation; so the way how this actions are performed depends on the hardware platform used. An illustration of a robot is shown in Figure 4.4.

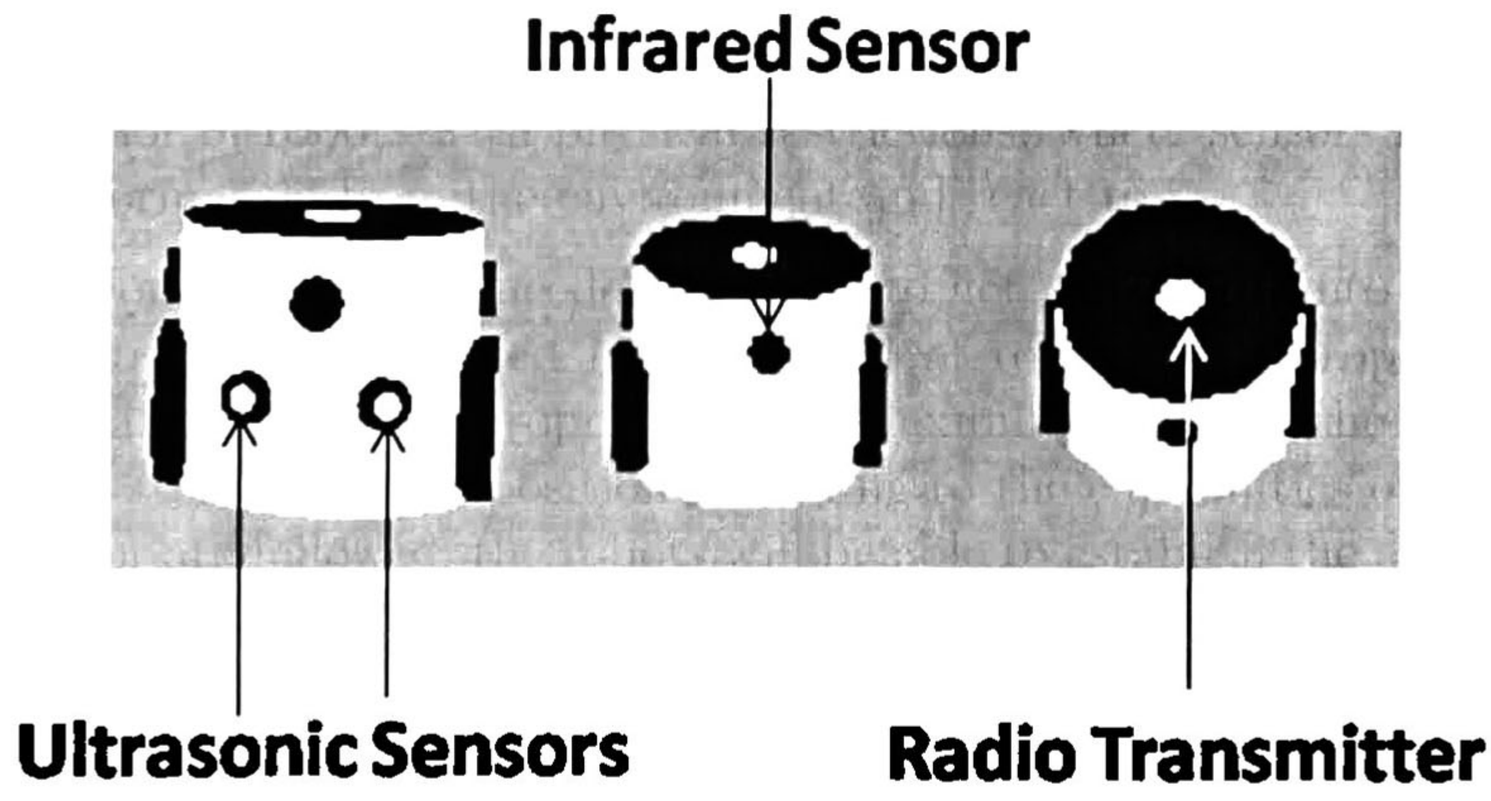



Figure 4.4: Robot Architecture.

The mobile platform is built around a circular base, and moves on two wheels that are independently driven. Robots have to be equipped with sensors to detect obstacles and beacons for robot search. Two ultrasonic sensors are arranged at the front of the robot. While the robot goes forward, the ultrasonic sensors allow a robot to avoid obstacles. An infrared sensor is fixed at every side of the robot. These sensors are used to achieve alignment between neighboring robots. A radio transmitter for sending and receiving messages is set

at the top of the robot. The signal strength of the radio transmitter from these sensor are used in order to invoke aggregation behavior.

The issues that a robot has to solve for getting its position in the formation are described below:

- *Find a robot.* Robots in formation send HELLO messages periodically. These messages are received by free robots. When a HELLO message is received, the robot detects the received signal strength. If the current strength is higher than the signal strength of the last message, then the robot follows the same course; however, if the signal strength is lower than the signal strength of the last message, then the robot tries to found another direction until a higher signal strength is detected. Through this behavior, the robots tend to move toward the robots who are in the formation.

- *Align with the neighbor.* At the moment of a free robot is near of a robot in the formation, the free robot initiates a message interchange in order to determine the position that the robot has to take in the formation. To determine the current robot position with respect to the neighbor in the formation, the ultrasonic sensors and the robot orientation are used. A rotation is performed by the robot until its ultrasonic sensors detect an object within a threshold distance. The current position (left, right, back, front) of the free robot is determined based on the orientation of the neighbor robot and the orientation of the free robot. Once the robot position has been determined, the free robot aligns with the neighbor using the infrared sensors.

- *Move to left.* When a robot has to move to the left and it is not the current position of the robot, it moves around the neighbor robot until the left position is reached. At this point, the robot moves away a distance $d$ in order to maintain the shape of the formation.

- *Move to right.* When a robot has to move to the right and it is not the current position of the robot, it moves around the neighbor robot until the right position is reached. At this point, the robot moves away a distance $d$ in order to maintain the shape of the formation.

### 4.2.2 Scenario Description

A set of seven robots were distributed randomly in a workspace of size 2m x 2m. A wedge formation is perfomed. Once the formation is established, the leader initiates a switching process in order to establish a line formation. Initially, a robot is defined as a leader. The rest of robots are distributed randomly in the environment (Figure 4.5).

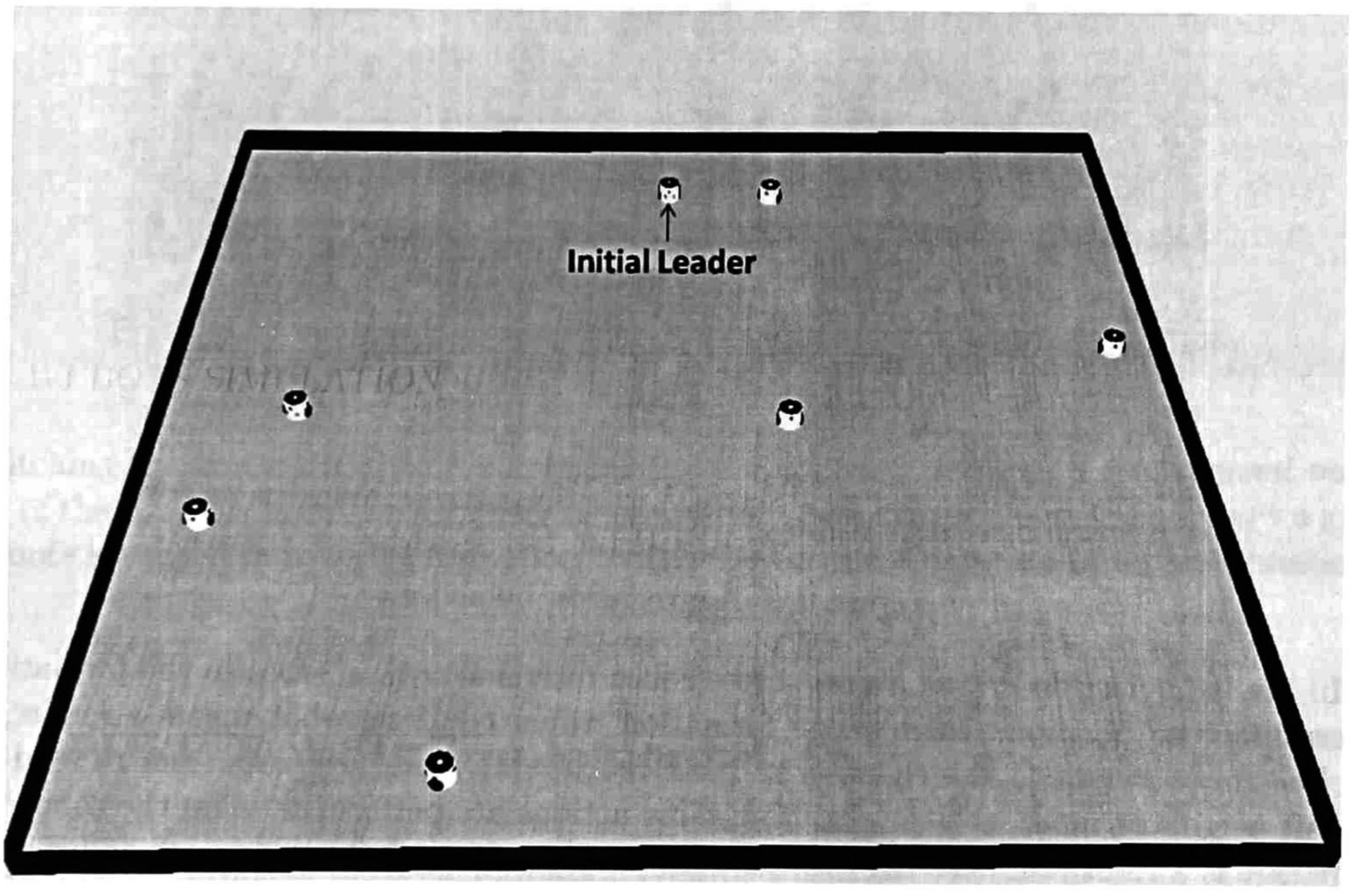



Figure 4.5: Robots distributed randomly in the workspace

The robots, which are in the formation, periodically send HELLO messages for allowing that free robots detect the formation. When a free robot receives a HELLO message, this robot tends to move toward the group of robots in the formation. When HELLO messages are not received, the free robot does not move. This aggregation behavior is illustrated in Figure 4.6.

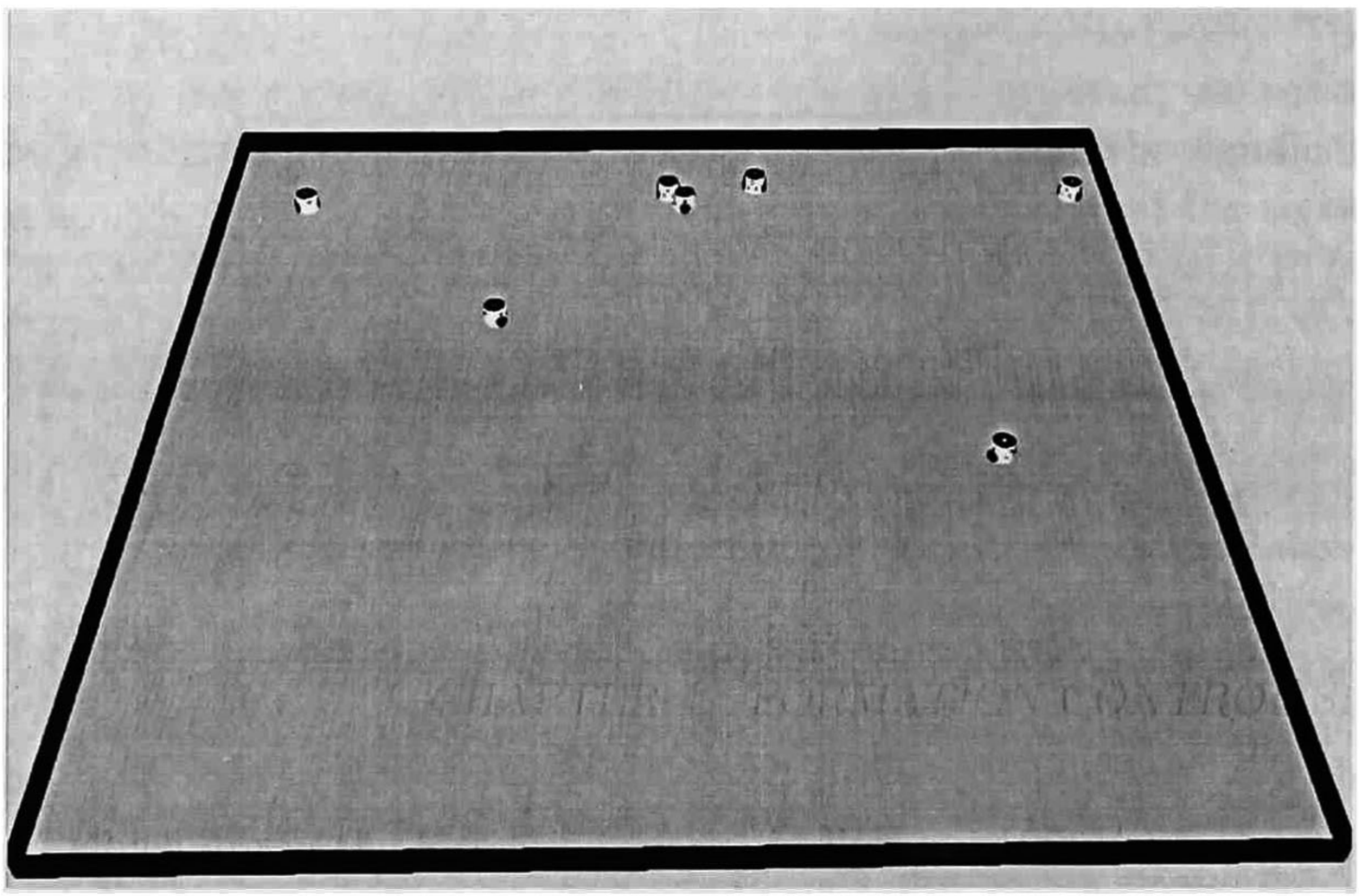


Figure 4.6: Aggregation behavior.

In the behaviors described above, a free robot interacts with a robot in the formation for determining its position. In a wedge formation, when the free robot meets a leader, then the free robot moves to the right. The right neighbors of the new robot go forward while the left neighbors move to the right. The same actions are performed when the robot in the formation is a right follower. On the other hand, if the free robot meets a left follower, then the free robot moves to the left. The right neighbors of the new robot go forward while the left neighbors move to the left. A robot being joined at the left side of the formation is shown in Figure 4.7.

Every time a new robot is added, a message is forwarded until the leader receives it. The leader maintains a count of new robots integrated in the left and the right sides. When the leader detects that there are two more robots in a side than the number of new robots in the other side, a balance maneuver is initiated and the leader role is reassigned. This role is assigned to the new robot in the middle of the formation. This process is illustrated in Figure 4.8.

In order to give to the formation the capability of adapting to the environment conditions or adapting to perform a certain task, the algorithm proposed allows to the leader to initiate

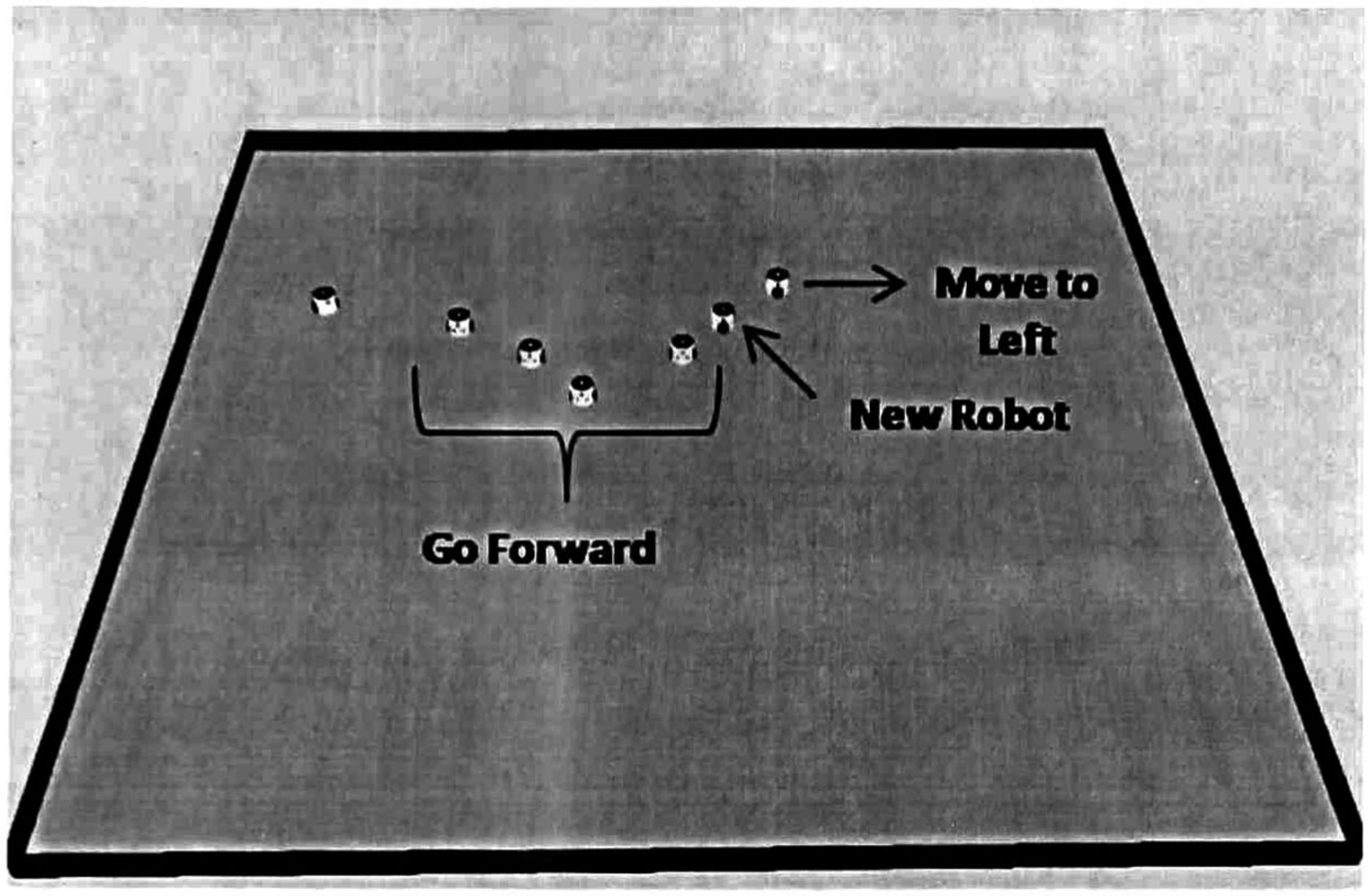


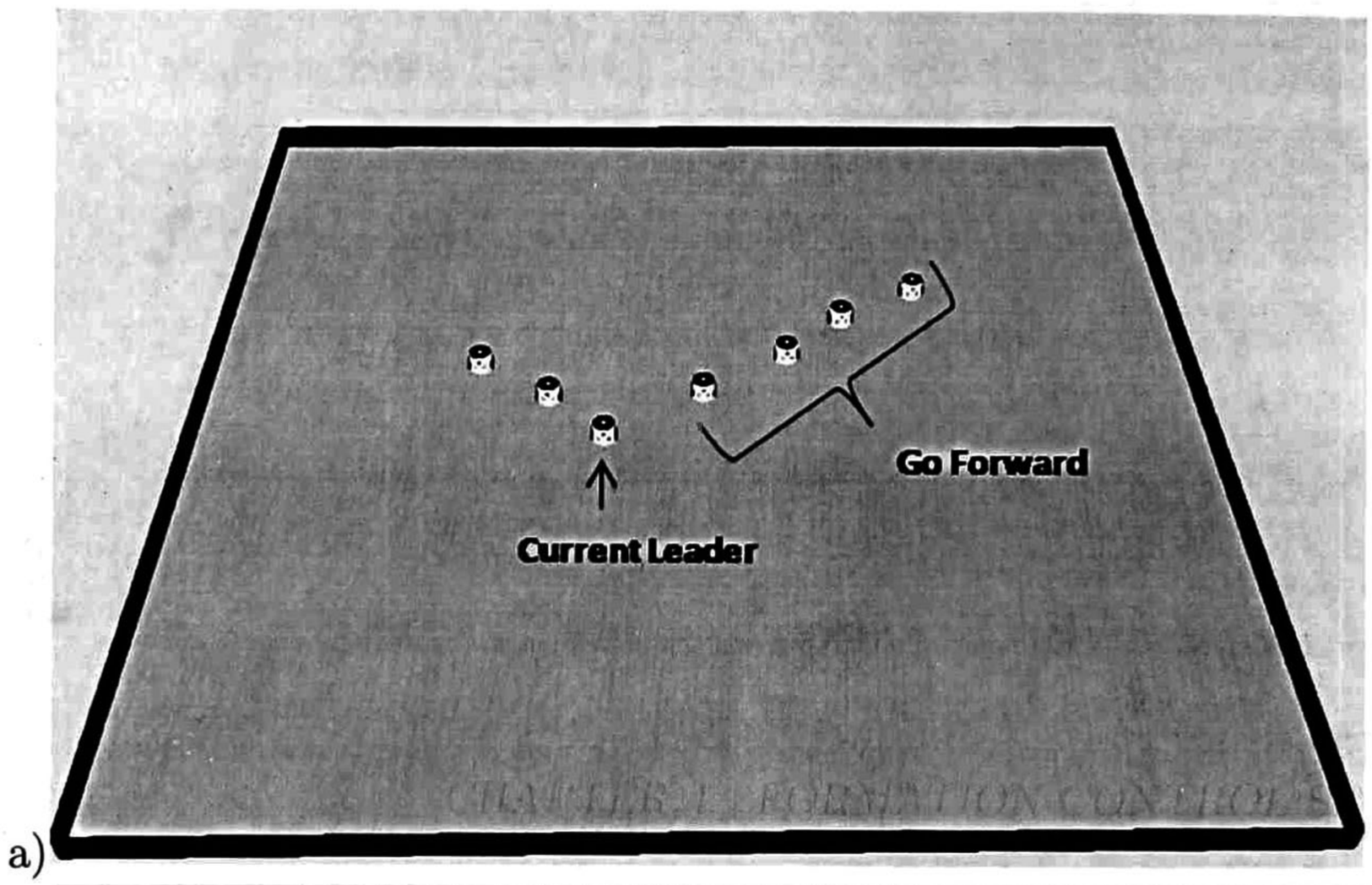

Figure 4.7: A new robot being joined at the left side of the formation.

a switching formation process. The action to be performed by every robot depend on the state of the robot. Possible actions were described in chapter 3. Figure 4.9 shows how a group of robots arranged in a wedge formation switch the current formation to a line formation.

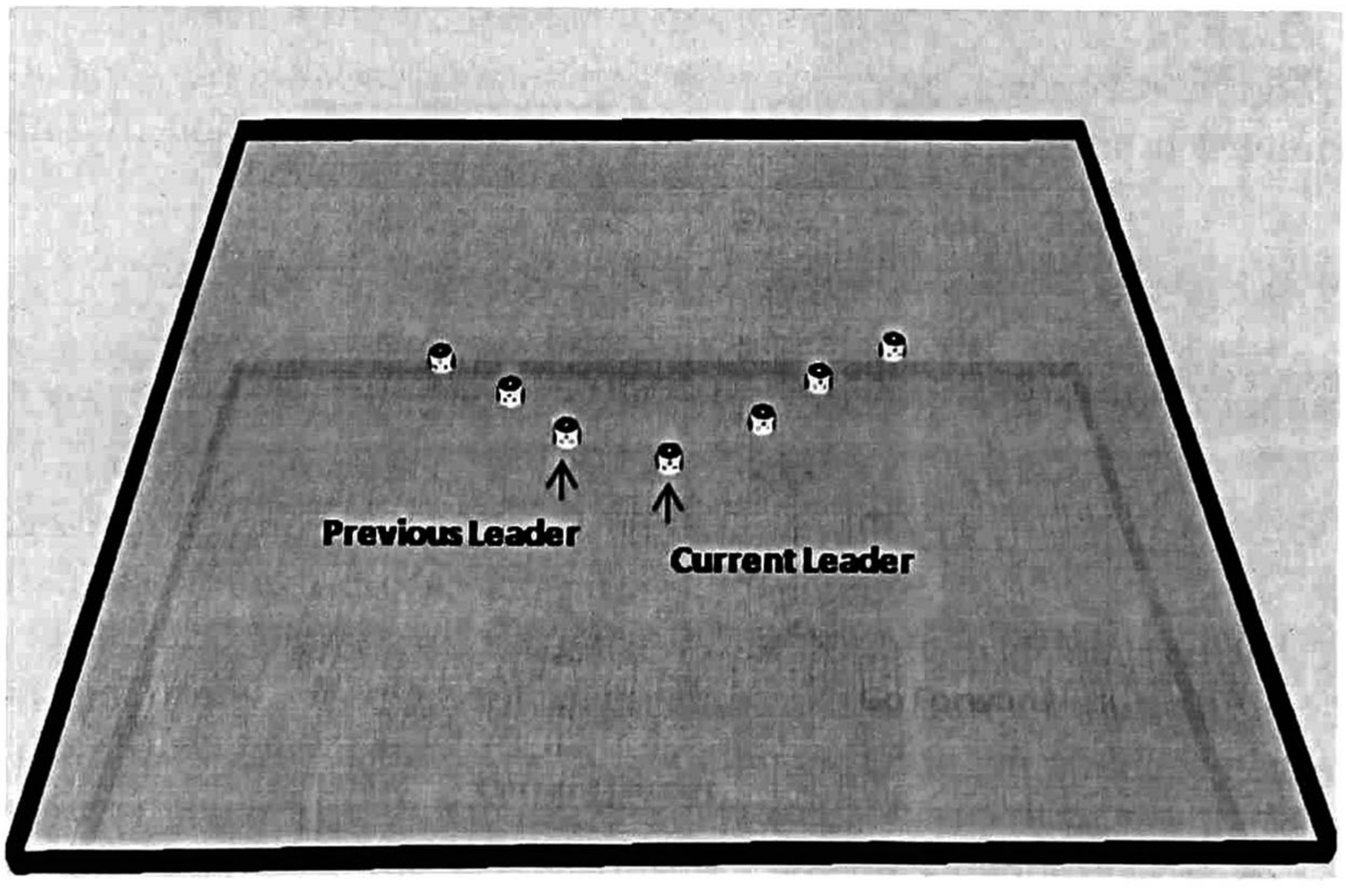
## 4.3 NXT Robots Implementation

For solving the problem of robot formation control with real robots, the Lego Mindostorms NXT Robots were used. Each robot has been designed to provide basic mobility and local interactions. Every robot is equipped with a compass sensor (the orientation of the robot is determined by this sensor), and a ultrasonic sensor (it helps to determine the distance with respect to other robots). In order to communicate between robots, the Bluetooth devices integrated in the robots are used. Two wheels with a radius of 56mm are used for allowing the robot to move in the environment. A third wheel of 24mm is used only for providing stability to the robot. A photograph of a robot is shown in Figure 4.10.

The wedge-formation algorithm is tested with a set of four robots. Among a series of experiments, several stages in the process of group formation task were described; in this stage, the actions performed by the robots can be observed. Figure 4.11 shows the snapshots taken of the experiment.

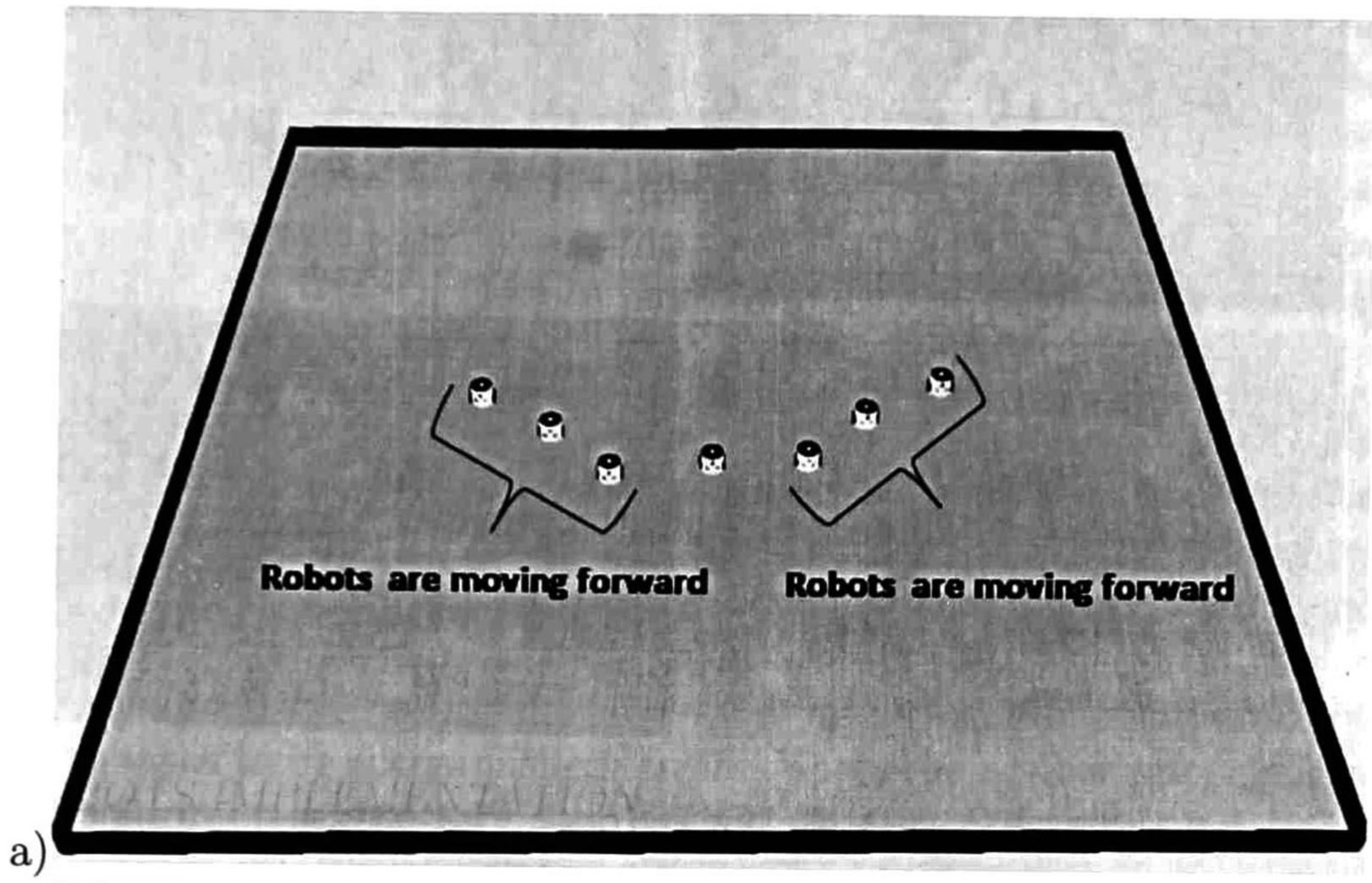



a)

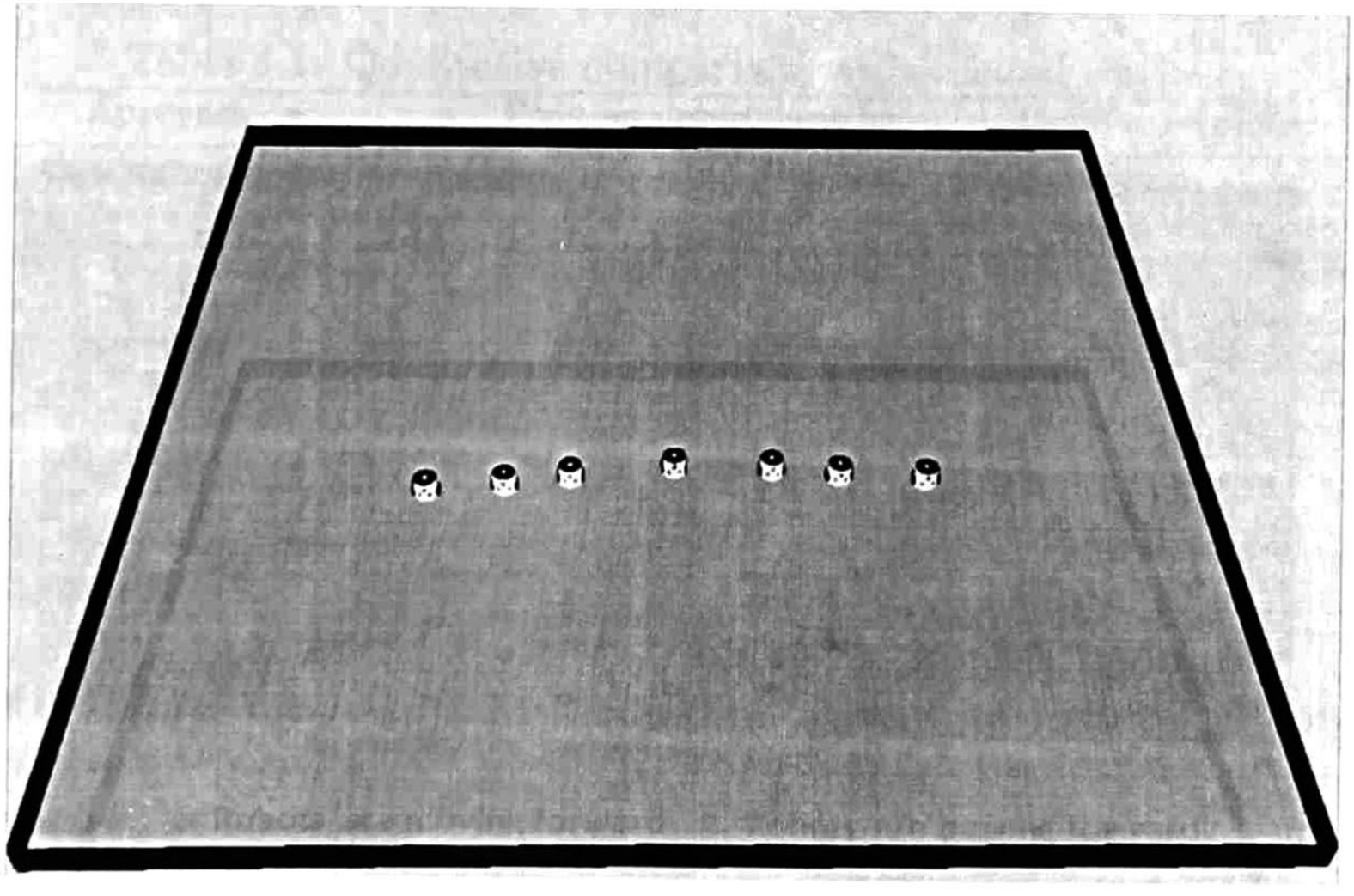



b)

Figure 4.8: a) Unbalanced Formation. b) Balanced Formation.




a)



b)

Figure 4.9: a) Switching Formation from wedge to line. b) Line Formation.

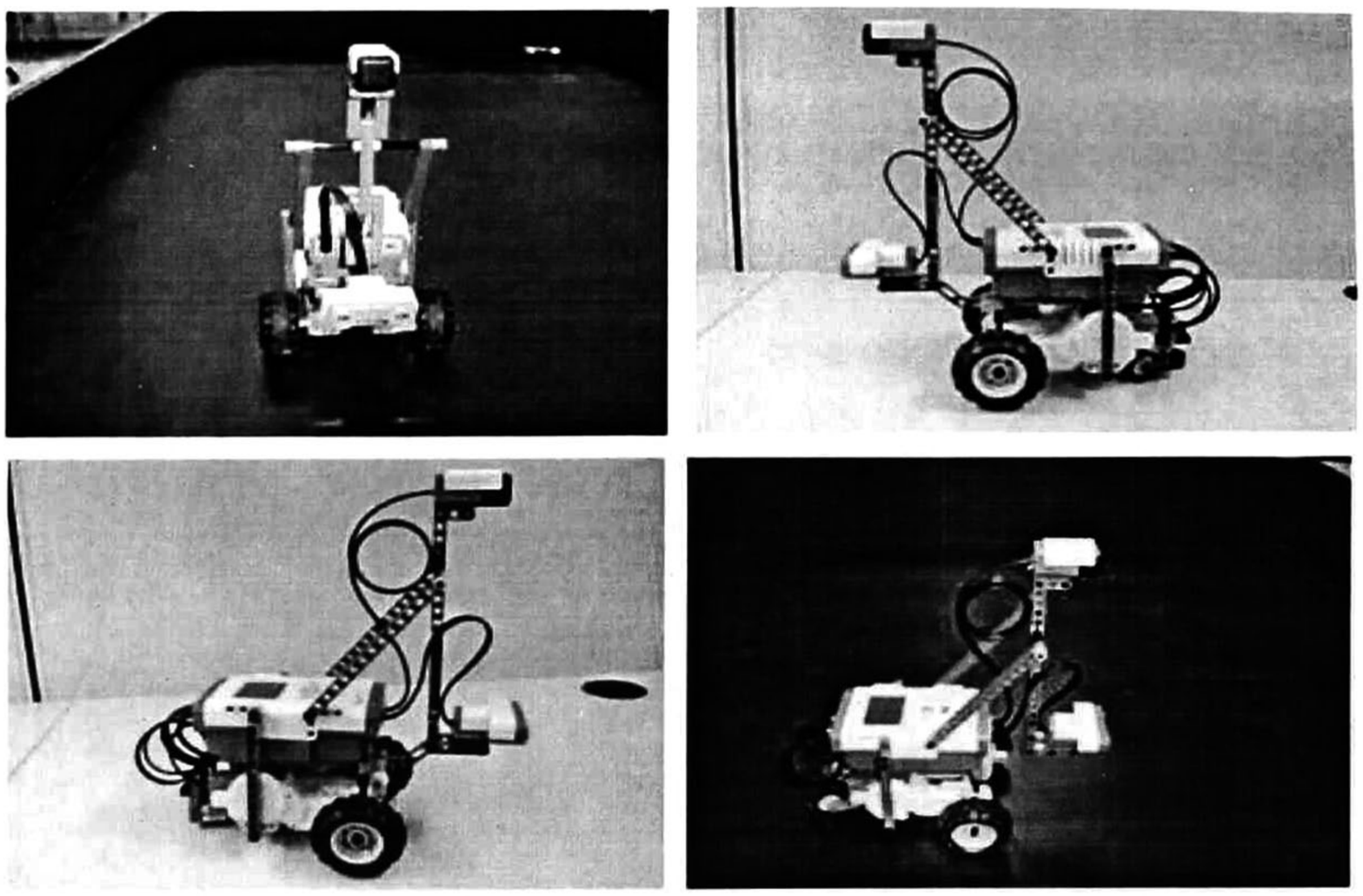


Figure 4.10: NXT Robot Platform.

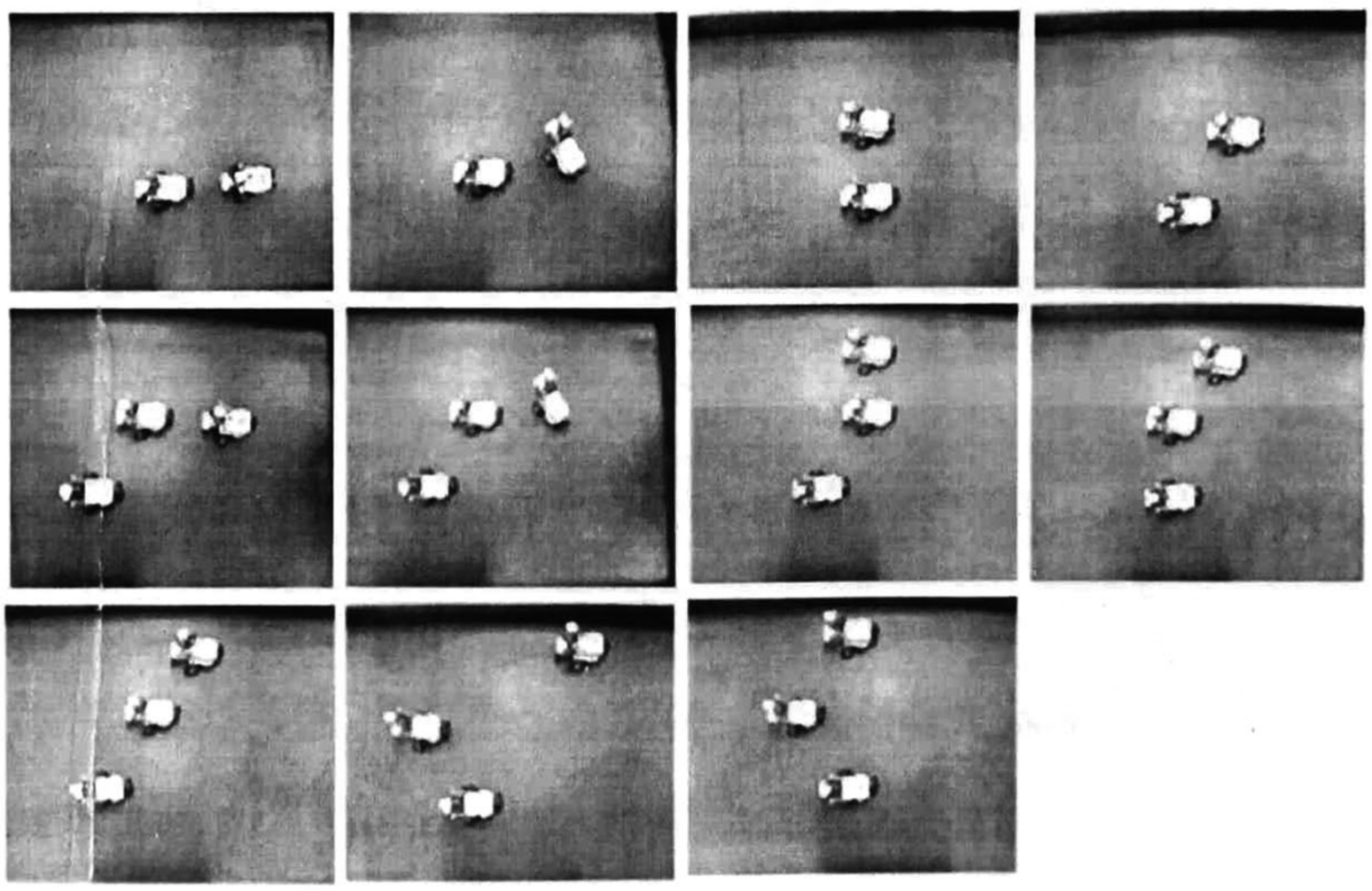


Figure 4.11: NXT implementation.

## 4.4 Simulation Results

Among a series of simulations that were conducted using NetLogo and Webots, it could be observed several stages of the self-organizing process in order to obtain the formation of the mobile robots. These stages include the initial step of the process, new robots being integrated in a formation, the balance maneuver, and finally the switching formation process. Through these simulations it can be observed how the local interactions between robots bring as a result the formation of the robots. According new robots are integrated in the formation, the formation is unbalanced. Then a balance maneuver is performed and the formation recovers its shape.

In order to compare the results obtained in this work with related works, the qualitative aspects of behaviors implemented in every work are considered. The quantitative aspects are not considered due that they depend on the hardware characteristics of robots used in every implementation. Furthermore, the algorithm proposed can be implemented in any platform which could be capable of performs the actions described in this work. The way on that actions are performed depends on the hardware characteristics, which can be elected according to the tasks to be accomplished. In this work, the actions are performed according to the robot platform presented here.

Table 4.1: Qualitative comparison with related works.

| Approach | Position determined by | Get the formation at |
|---|---|---|
| *At the dynamics level* | Keep only robot position | |
| *Using local information* | Leader | Target assigned by the leader |
| *Using local templates* | Neighbor state | Free atachment points |
| *Using self-organization principles* | Neighbor State | neighbor location |

| Approach | Balance Maneuver | Switching between Formations |
|---|---|---|
| *At the dynamics level* | Keep only robot position | |
| *Using local information* | Allowed | Not allowed |
| *Using local templates* | Not allowed | Not allowed |
| *Using self-organization principles* | Allowed | Allowed |

The works, which address the problem at the dynamics level, provide algorithms for keeping only the robot position. They do not consider the process to establish a formation before a robot knows its position. On the other hand, works using local information provide a solution for establishing a formation, but it is based on a little amount of global information, or exists a robot which decides the position of the other robots in the formation; furthermore, to switch between formations is not allowed. The approach, which uses local templates, avoids the use of global information. The formation is obtained through local interactions; however, balance maneuver and to switch between formations are not allowed.

The algorithm proposed here is extend for solving the drawbacks presented in the works

described above, while this algorithm considers self-organization principles in order to provide a scalable and adaptable approach.

# Chapter 5

# Conclusions and Future Work

## 5.1 Conclusion

In this thesis an algorithm for robot formation was presented. This algorithm is based on self-organization principles. The global behavior (robot formation) is obtained from local interactions. Every robot decides its position in the formation according to the information received from the neighbor robots. Furthermore, the robots do not have knowledge about the number of robots arranged into the formation. Since the robot behaviors depend on the state of the robot, new robots can be integrated to the formation without increasing the complexity of the algorithm; so, the algorithm presented maintains the scalability property. The algorithm has a linear complexity.

The formation is balanced when it is needed due to the insertion of new robots into the formation. The leader role is reassigned every time a balance maneuver is performed. Switching between formations gives the adaptability property to the group of robots arranged in the formation. With this behavior, robots can adapt to the environment conditions. The leader initiates the balance maneuver and the switching formation process. It is not yet decided autonomously by the robot when a switching process is required.

A Petri net model was presented for modeling the state of the robot. The behavior can be obtained autonomously from this model according to the messages received by the robot. The behavior resulting from the interactions between neighbors was simulated in NetLogo. The actions needed to get a position in the formation by a robot were described using Webots simulator. A robot architecture was proposed in order to achieve the required tasks. architecture was proposed in order to achieve the required tasks. architecture was proposed in order to achieve the required tasks. It was demonstrated through simulations how the robots, performing simple rules and local interactions, produce a desired global behavior.

## 5.2 Future Work

Several topics briefly mentioned in the text deserve further attention:

- In this work, the switching process is initiated by the leader when it is manually indicated. The algorithm devised here can be extended in order to decide autonomously when a switching process is required. This decision can be taken by considering the conditions of the environment or the task to be accomplished.

- As mentioned before, the Petri net can be used to auto-generate the code needed to perform the robot actions according to the received messages and the current status of the robot controller. This extension to the algorithm will provide a higher adaptability.

- In order to create a robust structure, an algorithm of obstacle avoidance, and algorithms for maintaining a distance with respect to other neighbors while the robots are moving, have to be integrated to the algorithm devised here.

# Bibliography


[1] http://www.cyberbotics.com.

[2] M. A. Sánchez Acevedo, E. López-Mellado, and F. Ramos-Corchado. Self organization algorithm for mobile devices. *In Encyclopedia of Information Science and Technology Second Edition*, 2008.

[3] M. A. Sánchez Acevedo, E. López-Mellado, and F. Ramos-Corchado. A self-organizing algorithm for mobile agents formation control. *International Journal Communications of SIWN 2008*, July 2008.

[4] I. Ayari and A. Chatti. Reactive control using behavior modelling of a mobile robot. *International Journal of Computers, Communications & Control*, 2(3):217–228, 2007.

[5] P. Bak. *How Nature Work: The Science of Self-Organized Criticality.* Springer-Verlag, September 1996.

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

[7] T. Balch and M. Hybinette. Social potentials for scalable multi-robot formations. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA '00*, 1:73–80, April 2000.

[8] A. Borshchev and A. Filippov. From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools. *22th International Conference of the System Dynamics Society*, July 2004.

[9] J. Cheng, W. Cheng, and R. Nagpal. Robust and self-repairing formation control for swarms of mobile agents. *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 1:59–64, July 2005.

[10] T. Dierks and S. Jagannathan. Control of nonholonomic mobile robot formations: Backstepping kinematics into dynamics. *Proceedings of 16th IEEE International Conference on Control Applications CCA*, 1:94–99, October 2007.

[11] K.D. Do and J. Pan. Nonlinear formation control of unicycle-type mobile robots. *Robotics and Autonomous Systems*, 55(3):191–204, March 2007.

[12] A. Fatmi, A. A. Yahmadi, L. Khriji, and N. Masmoudi. A fuzzy logic based navigation of a mobile robot. *Proceedings of World Academy of Science, Engineering and Technology*, 15:255–260, October 2006.

[13] J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, October 2002.

[14] J. Fredslund and M. J. Mataric. Robots in formation using local information. *Proceedings of 7th International Conference on Intelligent Autonomous Systems IAS-7*, 1:100–107, March 2002.

[15] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura. Self-organizing formation algorithm for active elements. *Proceedings of the 21st. IEEE Symposium on Reliable Distributed Systems*, 1:416–421, October 2002.

[16] F. Heylighen. *The science of self-organization and adaptivity in The Encyclopedia of Life Support Systems: Knowledge Management, Organizational Intelligence and Learning, and Complexity.* EOLSS Publishers, 1999.

[17] J. Hoffmann, M. Jüngel, and M. Lötzsch. A vision based system for goal-directed obstacle avoidance. *RobuCup - Lecture Notes in Computer Science*, 3276:418–425, 2005.

[18] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, May 1993.

[19] J. Kosecka and R. Bajcsy. Discrete event systems for autonomous mobile agents. *Robotics and Autonomous Systems*, 12(3-4):187–198, April 1994.

[20] K. N. Krishnanand and D. Ghose. Formations of minimalist mobile robots using local-templates and spatially distributed interactions. *Robotics and Autonomous Systems*, 53(3-4):194–213, December 2005.

[21] M. Lemay, F. Michaud, D. Létourneau, and J.-M. Valin. Autonomous initialization of robot formations. *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA '04*, 3:3018–3023, April 26 - 1 May 2004.

[22] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. *Proceedings of the 40th. IEEE Conference on Decision and Control*, 3:2968–2973, December 2001.

[23] M. López-Sánchez. Robot behavior adaptation for formation maintenance. *ICINCO-RA*, 1:283–288, August 2006.

[24] M. Mamei, M. Vasirani, and F. Zambonelli. Experiments of morphogenesis in swarms of simple mobile robots. *Applied Artificial Intelligence*, 18(9-10):903–919, October-December 2004.

[25] S. Mastellone, D. M. Stipanovic, and M. W. Spong. Multi-agent formation control and trajectory tracking via singular perturbation. *Proceedings of 16th IEEE International Conference on Control Applications CCA*, 1:557–562, October 2007.

[26] D. J. Naffin and G. S. Sukhatme. Negotiated formations. *Proceedings of the Eighth Conference on Intelligent Autonomous Systems*, 1:181–190, March 2004.

[27] S. Nefti, M. Oussalah, K. Djouani, and J. Pontnau. Intelligent adaptive mobile robot navigation. *Journal of Intelligent and Robotics Systems*, 30(4):311–329, 2001.

[28] H. V. D. Parunak and S. Brueckner. Entropy and self-organization in multi-agent systems. *Proceedings of the 5th International Conference on Autonomous Agents 2001*, 1:124–130, May 28 - 1 June 2001.

[29] H. H. Pattee. The complementarity principle in biological and social structures. *Journal of Social and Biological Structures*, 1:191–200, 1978.

[30] J. Pavón, J. Gómez-Sanz, A. Fernández-Caballero, and J. J. Valencia-Jiménez. Development of intelligent multisensor surveillance systems with agents. *Robotics and Autonomous Systems*, 55(12):892–903, December 2007.

[31] C. W. Reynolds. Flocks, herds, and schools: A distributed behavior model. *SIGGRAPH '87: Proceedings of 14th Annual Conference on Computer Graphics and Interactive Techniques*, 21(4):25–34, July 1987.

[32] W.-M. Shen, C.-M. Chuong, and P. Will. Simulating self-organization for multi-robot systems. *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and System.*, 3:2776–2781, September 30-4 October 2002.

[33] W. M. Spears, R. Heil, and D. Zarzhitsky. Artificial physics for mobile robot formations. *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics, SMC*, 3:2287–2292, October 2005.

[34] U. Wilensky. http://ccl.northwestern.edu/netlogo/. *Center for Connected Learning and Computer-Based Modeling Northwestern University, Evanston, IL*, 1999.

[35] D. Zhang, G. Xie, J. Yu, and L. Wang. Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, 55(7):572–588, July 2007.

# CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
# UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

**Un algoritmo de auto-organización para la formación de agentes móviles**

del (la) C.

**Miguel Angel SÁNCHEZ ACEVEDO**

el día 15 de Agosto de 2008.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Federico Sandoval Ibarra
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara