sensors

MDPI

Article

UAV Swarms Behavior Modeling Using Tracking Bigraphical Reactive Systems

Piotr Cybulski * and Zbigniew Zieliński

Faculty of Cybernetics, Military University of Technology, ul. gen. S. Kaliskiego 2, 00-908 Warsaw, Poland; zbigniew.zielinski@wat.edu.pl

* Correspondence: piotr.cybulski@wat.edu.pl

Abstract: Recently, there has been a fairly rapid increase in interest in the use of UAV swarms both in civilian and military operations. This is mainly due to relatively low cost, greater flexibility, and increasing efficiency of swarms themselves. However, in order to efficiently operate a swarm of UAVs, it is necessary to address the various autonomous behaviors of its constituent elements, to achieve cooperation and suitability to complex scenarios. In order to do so, a novel method for modeling UAV swarm missions and determining behavior for the swarm elements was developed. The proposed method is based on bigraphs with tracking for modeling different tasks and agents activities related to the UAV swarm mission. The key finding of the study is the algorithm for determining all possible behavior policies for swarm elements achieving the objective of the mission within certain assumptions. The design method is scalable, highly automated, and problem-agnostic, which allows to incorporate it in solving different kinds of swarm tasks. Additionally, it separates the mission modeling stage from behavior determining thus allowing new algorithms to be used in the future. Two simulation case studies are presented to demonstrate how the design process deals with typical aspects of a UAV swarm mission.

Keywords: modeling; bigraphs; unmanned aerial vehicles; UAVs; swarm; planning; agent behavior; swarm robotics; multi-agent systems

check for updates

Citation: Cybulski, P.; Zieliński, Z. UAV Swarms Behavior Modeling Using Tracking Bigraphical Reactive Systems. *Sensors* **2021**, *21*, 622. https://doi.org/10.3390/s21020622

Received: 13 December 2020
Accepted: 12 January 2021
Published: 17 January 2021

1. Introduction

Recently, unmanned aerial vehicles (UAVs) have been increasingly used both in the civilian and military spheres, mainly due to their relatively low cost, flexibility, and the elimination of the need for on-board pilot support. The use of UAV swarms is of particular importance, especially with increased autonomy of its elements. It is expected [1] that autonomous UAV swarms will become a key element of future military operations, as well as civilian applications including security, reconnaissance, intrusion detection, and support Search and Rescue (SAR) or Disaster Recovery (DR) operations. DR operations are extremely challenging, and in the immediate aftermath of a disaster, one of the most pressing requirements is for situational awareness. UAV swarms provide an indispensable platform for building the situation awareness in such cases. The obvious benefit of using UAV swarms is an increase in the efficiency of the operation, an accelerated process of its execution and an increased probability of success. Their use in wilderness search and rescue (WiSAR), in particular, has been investigated for fast search-area coverage. One of the most important task in WiSAR is search – until a missing person has been found, they cannot be rescued or recovered. Many search tasks require a number of UAVs to remain in communication at all times and in contact with the base station via a short range ad hoc wireless network. For example, a swarm of UAVs must disperse (take the proper starting positions) to find the missing person as quickly as possible before their energy reserves run out. However, in order to efficiently operate a swarm of UAVs, it is necessary to address the various autonomous behaviors of its constituent elements, sometimes even with conflicting goals, to achieve a high level of adaptation and human-like cognitive behavior. Therefore, it is necessary to conduct research on methods of increasing the autonomy and interoperability of UAVs while limiting global communication and human dependence on the operator.

An individual UAV can perform different tasks, such as terrain reconnaissance, close-up inspection of selected areas, transfer of communication, and target pursuit. The UAV can intelligently take over the role depending on the situation. Such a high degree of adaptation and cooperation in complex scenarios requires innovative solutions at the design stage of the UAV swarm system and appropriate methods of its verification and testing.

A UAV swarm is a special case of robot swarm. There are a few definitions of *robot swarm*, most of them differ in capabilities of its elements and a swarm itself but all of them link it to multirobot systems. A *multirobot system* consists of multiple robots cooperating with a goal to accomplish a given task. The main features associated to multirobot systems are scalability, robustness, flexibility, and decentralized control. In [2], multirobot systems that are not swarms are defined as those that have explicitly stated goals and in which robots are executing individual and/or group tasks. Additionally, robots in such systems have roles that can change during a course of a mission. In the same work, it was pointed out that in swarm system a swarm behavior emerges from local interactions between robots. In [3], the authors defined as a swarm system any robotic system that is capable of performing "swarm behavior". A frequently quoted definition of robotic swarms is the one presented in [4]:

> *"Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment."*

In the same paper, it was recommended for a swarm system to has the following properties.

- Robots in a swarm should be autonomous and have the abilities to relocate and interact with other objects in the environment.
- A swarm control method should allow for coexistence of large number of robots.
- A swarm can be either homogeneous or heterogeneous. If a swarm is heterogeneous, it consists of multiple homogeneous subgroups.
- Communication and perceiving capabilities are local. It means that robots do not know a global state of the environment at any moment.

For the purpose of this article, we will adopt the definition presented in [3] with the features described above.

In recent surveys [2,3,5], there are multiple classifications of tasks that a swarm can be given to perform. Below, we will use one that was defined in [5].

Swarm tasks can be divided in three categories:

1. Spatial organization—tasks associated with this category focus on obtaining some spatial property by a swarm. An example of such property might be distance between robots. Typical tasks of this kind are *aggregation*, *dispersion*, *coverage*, and *pattern formation*.
2. Collective motion—this group consists of obstacle avoidance and objects gathering tasks. What makes collective motion different from spatial organization is that in the latter we are mainly focused on rearrangement of individual robots within a swarm while in collective motion we are generally focused on swarm as a whole. Typical tasks in this group are *exploration*, *foraging* (finding and collecting specific objects on the map), *collective navigation* (it aims at constructing, maintaining and, if there is such a need, modifying a formation heading toward some direction) and *collective transport* (in which a swarm tries to move an object that is otherwise too heavy for a single robot in the swarm).

3. Decision-making—in this group robots make a decision that should lead to a consensus within a swarm. The decision is based on the local perception of the environment and information received from other robots. In the context of robot swarms, this kind of task appears in situations where there is no access to globally shared information. Typical tasks in this group are *consensus* (where a swarm tries to settle on a decision that every of its members agree on), *task allocation* (where robots select from an array of available tasks to perform), and *localization*.

Due to the large number of tasks that a swarm can be given to perform and their complexity, many areas were inspirations for swarm robotics over the course of years. Based on the survey presented in [2], we can distinguish four main areas that served as an inspiration for robot swarms design:

- Biology—a vast number of solutions and design methods originated directly from observation of real world swarms. To name a few, birds flocks, bees, and ants swarms served as such source of inspiration. A well-known example of robotic swarm originated from biology is presented in [6]. All solutions based on evolutionary processes can be also included in this category. A complex introduction to bioinspired multirobot systems can be found in [7,8].
- Control theory—this category includes all designs where physical aspects of robots are modeled as continuous-time continuous-space dynamical system and communication between robots is modeled using graph theory. In some works [3], designs based on graph-theory are considered as a separate group. A concise introduction to solutions extensively using control theory can be found in [9]. What is worth noting is that these kinds of designs methods give formal guarantee of correct execution as long as the requirements are met. Unfortunately, this group poorly takes into account indeterministic mission elements and requirements for a swarm are often unrealistic, as it was stated in [10].
- Amorphous computing and aggregated programming—the main idea behind amorphous computing [11] is to use a large number of identical computers distributed across a space. It is assumed that these computers have only local communication capabilities and do not know their position. Because of its assumptions amorphous computing closely reassembles swarm systems. An example of software implementation of this paradigm is Proto language [12]. In turn, aggregated programming [13] is a paradigm that focuses on the development of large-scale systems from the perspective of their totality rather than individual elements. One prominent aggregate programming approach is based on the field calculus [14]. An implementation of this parading is, for example, Protelis [15]. It is worth noting that it currently used to model IoT-like systems.
- Physics—swarm design methods inspired by physics are mainly focused on two ideas: artificial forces [16] and Brownian motion [10]. As it was pointed out in [2], a characteristic feature of physics-inspired swarm design methods is that they tend to consider interactions between robots as passive. It means, there may be no message-exchanging communication between agents, instead robots can interact indirectly with each other (most of the time using some kind of forces).

There are multiple taxonomies concerning different aspects of swarm robotics. This includes swarm design methods and methods of analysis of both models and swarms themselves. For example, the taxonomy proposed in [17] distinguishes swarms based on their features, such as their size or communication capabilities. Another taxonomies presented in [18,19] categorize, among others, methods of swarm modeling and its analysis as well as different ways to design its behavior. These taxonomies are especially important for us so we can compare our proposition to the existing methods of swarm design. In [18], the authors divided methods of swarm modeling into two groups. Fist group, called *top-down* (sometimes referred to as macroscopic methods) encapsulate all methods that start from defining a desired swarm behavior and then try to construct robots that exhibit this behavior. The second way to designing robotic swarms, defined as *bottom-up* or microscopic, focuses first on capabilities and behavior of members of a designed swarm. Next, it is checked if the designed swarm is capable of carrying out a given mission. Both design methods have their pros and cons as it was discussed in [20]. The key difference between both methods is where does a design method start from.

In the same work, swarms been distinguished based on their capabilities to improve results. These can be either *non-adaptive*, *learning*, or *evolutionary*. A swarm is *non-adaptive* if the only way to improve its performance is by manual modification by the designer. In turn, a swarm can be described as *learning* if parameters of an algorithm it is using are automatically modified during task execution. Finally, if these parameters are modified in an iterative manner during the design stage with a use of evolution-based techniques, we can describe swarm behavior as *evolving*. In [19], a similar classification of swarm design methods have been proposed. According to this taxonomy, design methods can be described either as *behavior-based* or *automatic*. The first group consists of all methods where a swarm behavior is designed manually by the designer and improved with the trial and error method. The second group is made of all methods where a swarm behavior is constructed without a substantial involvement of the designer.

A constructed swarm model with a behavior policy for the swarm elements can be verified in two ways: using real robots or with simulators. This work is focused on earlier stages of robotic swarm development so we will only briefly cover the key achievements in this field.

The most obvious way to verify a robotic swarm model is with real robots. The most commonly cited swarm robots projects are *swarm-bots* [21], its successor *swarmanoid* [22], and the *Kilobot* project [23]. All of them are capable of performing multiple types of swarm behavior, which suggests that they are all equipped with sufficiently powerful hardware. This, in turn, let us to believe that the lack of common use of robotic swarms is due to insufficient behavior modeling techniques.

Based on an up-to-date state-of-the-art survey [5], it can be seen that there is a number of different simulators designed to help designers verify their work. They vary in terms of performance and versatility of accepted solutions. To name a few, in our opinion two of them are worth to recommend for those wanting to verify their theoretical results:

- ARGoS [24]—is an open source simulator, whose key features are efficiency, flexibility, and accuracy. According to the information provided by the author, it is used by academic community around the world.
- CoppeliaSim [25]—(previously known as V-REP) is a very advanced simulator which seems to be used by many commercial and academic institutions globally. It is free for academic use.

One of the proven methods of designing complex systems, which UAV swarm systems certainly are, is engineering based on formal models. Formal models offer a number of possibilities to automate the system design process, including verification of the behavior of the designed system. They allow us to better understand and facilitate analysis of a modeled system. Formal models provide mathematical abstractions of the designed system and can be validated against requirements, tested using various infrastructures, and can also be used to directly simulate the behavior of the system. One of such formalisms which can be used for UAV swarms modeling are bigraphs with tracking. Bigraphs were introduced by R. Milner [26] as a formalism to model systems in which placement and intercommunication between elements play an important role. Despite its novelty, there are already few extensions that allow to broaden its applicability. These are, among others, stochastic bigraphs [27], bigraphs with sharing [28] or bigraphs with tracking [26]. A quick introduction to bigraphs with a real-world use case can be found in [29].

It is important to emphasize that there are currently very few works on robot swarms using bigraphs. Examples [30,31] in the field of multi-agent system do not typically show how to generate behavior policy for swarm elements based on created models. The only solution we have found that does present a method of generating a behavior policy based on bigraphical model was presented in [32]. It uses a basic bigraphical notation mixed with actors model [33]. In our opinion, it is not an automatable method of swarm design.

Currently, there are only few tools supporting design with bigraphs, although it seems there are ongoing works [34] to change that. To our best knowledge, there are only two utilities for designing with bigraphs that are beyond proof-of-concept stage. The most advanced tool for modeling, verifying and simulation of bigraphical system *BigraphER* [35]. The second one, a tool for verification of reachability of states *Bigraphical Model Checker* (*BigMC*) [36] is no longer developed.

In this paper, we will present a method of modeling a UAV swarm with the addition of generating a behavior policy for swarm elements based on constructed model. Our goal is to present a swarm modeling method with the following features:

- It separates modeling stage from generation of behavior for swarm elements.
- It is flexible in the meaning that it can be used for a large number of different swarm tasks.
- It is capable of generating behavior policies on multiple levels of abstractions (from a single agent, through their groups, to an entire swarm as a whole).
- It is highly automatable. It is a desirable property because it indirectly enforces universal applicability of a method to different design problems. Additionally, automatic methods that are not monolithic tend to be modular, this in turn leads to standardization.

In the next section, we will present a method of modeling UAV swarm systems based on bigraphs with tracking. We will also define a way of constructing behavior policy which will guarantee a successful carrying out a given mission, assuming requirements that had been previously defined are met. Our method is inspired by the work presented in [37]. Although very interesting, it has two major shortcomings. First, the requirement definition stage is loosely coupled with the modeling stage. We wanted to address this issue and allow to formally transform capabilities of robots and mission requirements into model elements. The second issue is the assumption of identical behavior for all swarm elements. We do not consider this a necessary requirement for a swarm, although it may differ depending on the accepted definition of robot swarm.

One of advantages of our method is that the whole process can be automated from the moment of defining mission requirements (as bigraphical patterns) and robots capabilities. We have proved it with software libraries [38–40].

To summarize, according to taxonomies presented in [18], our method can be categorized as *bottom-up*, problem-agnostic and a generated behavior can be considered as *non-adaptive*. In turn, using the taxonomies presented in [19], our method can be considered as *automatic* and a method of analysis of a constructed model can be viewed as *macroscopic* (i.e., we are analyzing whole swarm and not individual interactions between its elements).

2. Methods and Materials

In this section, we will define formal elements and operations necessary to model a UAV swarm mission and determining a sequence of actions for the swarm elements. We have provided micro-examples at the end of each subsection for easier understanding.

Our proposition can be described as follows. We start from defining a UAV swarm mission as Tracking Bigraphical Reactive System (TBRS). We then transform this TBRS into state space represented as directed multigraph. Finally, we construct a behavior policy for swarm elements. As we treat a state space as a directed multigraph with edges corresponding to actions performed by swarm elements, we can define behavior policy as a walk (a finite length alternating sequence of vertices and edges) from the vertex representing the initial state of the mission to a vertex representing a final state (there can be few of those). A final state is a desirable outcome of the mission. We have proposed a method of finding all walks between any pair of vertices consisting of specified number of edges or loops.

2.1. Bigraphs

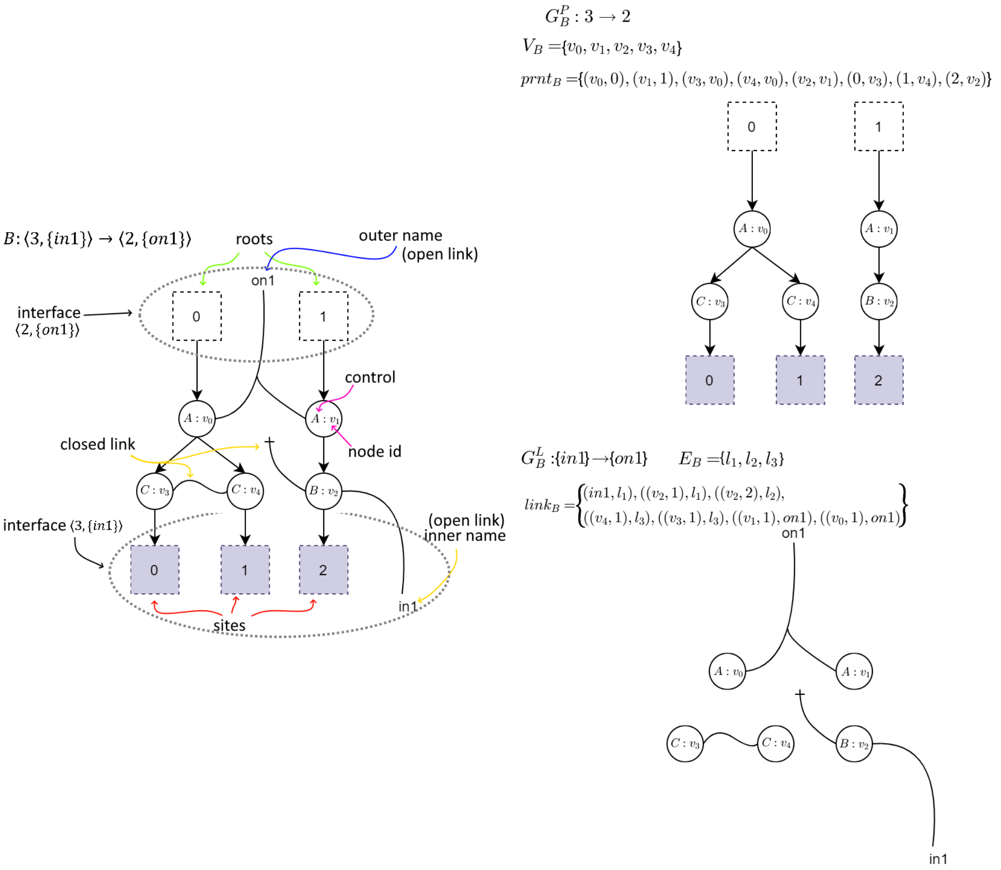
A bigraph consists of two graphs: a place graph and a link graph. Place graph is intended to model spatial relations between system's elements. Link graph is a hypergraph that can be used to model interlinking between the elements.

Formally a bigraph is defined as

$$B = (V_B, E_B, ctrl_B, G_B^P, G_B^L) : I \rightarrow O$$

- V_B—a set of vertices identifiers;
- E_B—a set of hyperedges identifiers;
- $ctrl_B$: $V_B \rightarrow K$— a function assigning a control type to vertices. K denotes a set of control types and is called a signature of the bigraph;
- $G_B^P = \langle V_B, ctrl_B, prnt_B \rangle : m \rightarrow n$ and $G_B^L = \langle V_B, E_B, ctrl_B, link_B \rangle : X \rightarrow Y$ denote a place and a link graph, respectively. A $prnt_B$ function defines hierarchical relations between vertices, roots, and sites. A $link_B$ function defines linking between vertices and hyperedges in the link graph;
- $I = \langle m, X \rangle$ and $O\langle n, Y \rangle$ denote an inner face and outer face of the bigraph B. By m, n we will denote a set of preceding ordinals of the form: $m = \{0, \ldots, m-1\}$. Sets X and Y represent inner and outer names, respectively.

A graphical example of a bigraph is presented in Figure 1.



B: $\langle 3, \{in1\} \rangle \rightarrow \langle 2, \{on1\} \rangle$
roots
outer name
(open link)
on1
interface
$\langle 2, \{on1\} \rangle$
0
1
control
$A : v_0$
$A : v_1$
closed link
node id
$C : v_3$
$C : v_4$
$B : v_2$
interface $\langle 3, \{in1\} \rangle$
(open link)
inner name
0
1
2
in1
sites
$G_B^P : 3 \rightarrow 2$
$V_B = \{v_0, v_1, v_2, v_3, v_4\}$
$prnt_B = \{(v_0, 0), (v_1, 1), (v_3, v_0), (v_4, v_0), (v_2, v_1), (0, v_3), (1, v_4), (2, v_2)\}$
0
1
$A : v_0$
$A : v_1$
$C : v_3$
$C : v_4$
$B : v_2$
0
1
2
$G_B^L : \{in1\} \rightarrow \{on1\}$
$E_B = \{l_1, l_2, l_3\}$
$link_B = \left\{ \begin{matrix} (in1, l_1), ((v_2, 1), l_1), ((v_2, 2), l_2), \\ ((v_4, 1), l_3), ((v_3, 1), l_3), ((v_1, 1), on1), ((v_0, 1), on1) \end{matrix} \right\}$
on1
$A : v_0$
$A : v_1$
$C : v_3$
$C : v_4$
$B : v_2$
in1

(**a**) A bigraph (**b**) A place graph and a link graph

Figure 1. An example of a bigraph and its constituents. The right part represents a place graph (the upper part of the figure) and a link graph (the lower part of the figure). They share a signature which defines control types (letters in nodes) and arity of each control (number of unique links that can be connected to a node with specified control). On the left there is the bigraph made from the superposition of them both.

Reaction rules are used to model dynamics in bigraphical systems. In this paper, we will use simplified tracking reaction rules. We call them simplified because only vertices will be tracked between reactions, as opposed to the original bigraphs with tracking proposed by Milner [26], where both vertices and hyperedges were tracked between reactions. Informally, a reaction rule defines a pattern (redex) in a source bigraph that shall be replaced with another bigraph (reactum). We will omit how patterns are found in bigraphs and how replacement is being done.

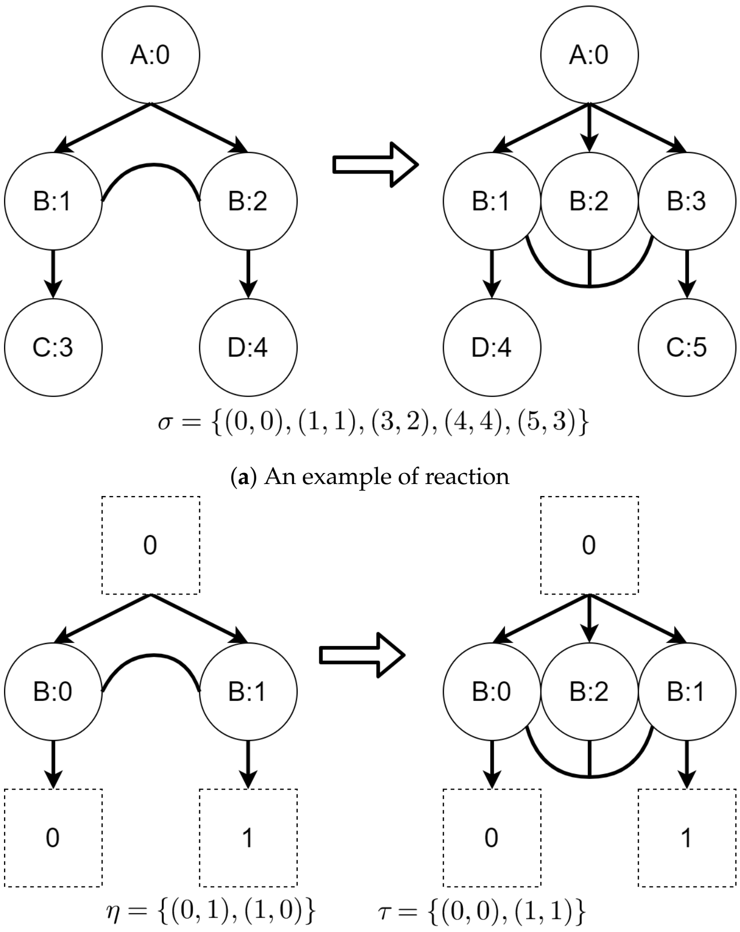
Formally, a tracking reaction rule is a quadruple:

$$(B_{redex} : m \to I, B_{reactum} : m^{'} \to I, \eta, \tau),$$

where

- B_{redex}—redex (a bigraph-pattern to be found in a bigraph to which rule is applied);
- $B_{reactum}$— reactum (a bigraph replacing redex);
- $\eta : m^{'} \to m$—a map of sites from reactum to redex;
- $\tau : V_{reactum} \to V_{redex}$—a partial map of reactum support onto redex support. It allows to indicate which elements are "residues" of a source bigraph in an output bigraph.

An example of reaction rule and its application is presented in Figure 2. A σ function denotes a residue of a source bigraph in an output bigraph.



$\sigma = \{(0,0),(1,1),(3,2),(4,4),(5,3)\}$

(**a**) An example of reaction

$\eta = \{(0,1),(1,0)\}$ $\tau = \{(0,0),(1,1)\}$

(**b**) An example of reaction rule

Figure 2. An example of bigraphical reaction with a corresponding reaction rule. The switch of vertices with controls *C* and *D* is caused by the η function. The σ mapping denotes which vertex in the output bigraph corresponds to which vertex in the source bigraph. It shows that the vertex with id 2 is "new" (it is not a residue of the source bigraph).

Having defined the bigraphical reaction rules, we can proceed to the definition of *Bigraphical Reactive System* (BRS). A BRS is a tuple $(\mathcal{B}, \mathcal{R})$ where $\mathcal{B}$ denotes a set of bigraphs with empty inner face and $\mathcal{R}$ is a set of reaction rules defined over $\mathcal{B}$. If $\mathcal{R}$ consists of rules with tracking then a pair $(\mathcal{B}, \mathcal{R})$ makes a *Tracking Bigraphical Reactive System* (TBRS).

Having a BRS we can generate a Transition System. A *Transition System* is a quadruple: $\mathcal{L}$ = (Agt, Lab, Apl, Tra), where

- Agt—a set of agents (i.e., bigraphs with an empty inner face, denoted as ϵ);
- Lab—a set of labels;
- Apl $\subseteq$ Agt × Lab—an applicability relation;
- Tra $\subseteq$ Apl × Agt—a transition relation;

For the purposes of this work, we will define a *Tracking Transition System* (TTS) $\mathcal{L}_{\mathcal{T}}$ = (Agt, Lab, Apl, Par, Res, Tra). First, three elements have the same definition as described above, the rest is defined as follows.

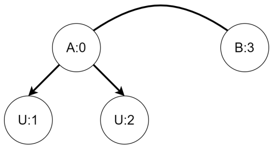
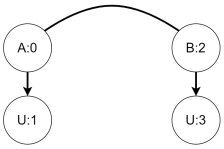
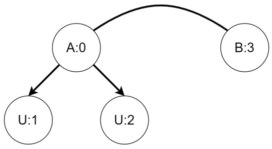
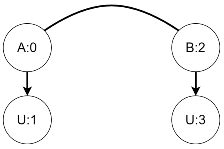
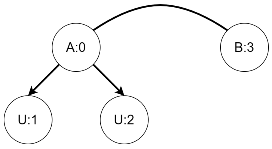
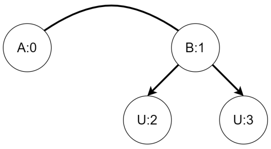
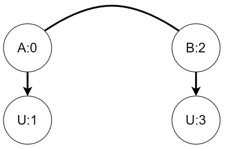
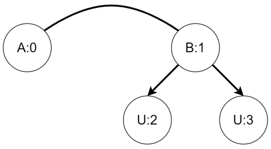
- $\mathrm{Par}(b, l) = p \quad b \in \mathrm{Agt}, l \in \mathrm{Lab}, p \in 2^{V_b}$—a participation function. It indicates which elements of a source bigraph participate in a transition. To avoid ambiguity, *Par* function should return an injective mapping between redex's support of the reaction rule corresponding to the transition's label and the source bigraph of the transition. We have omitted this in the definitions for the sake of simplicity but the implementation provided in [38] includes this in an output. The definition of the *Par* function provided in this paper allows us to indicate who is participating in a transition but does not indicate what role a participant takes.
- $\mathrm{Res}(b_1, l, b_2) = \sigma(b_2) \quad b_1, b_2 \in \mathrm{Agt}, l \in \mathrm{Lab}$—a residue function. It maps vertices in an output bigraph that are residue of a source bigraph to the vertices in the source bigraph;
- Tra $\subseteq$ Apl × Agt × Par × Res—a transition relation.

A Tracking Bigraphical Reactive System can be transformed into a Tracking Transition System.

A micro-example of Tracking Transition System is presented in Table 1. Each row describes a single transition in the system. The initial state of the system is presented in the first row in the first *Agt* column. The scenario that this TTS models is as follows. Two UAVs denoted as nodes with controls of type *U* are trying to move from an area of type *A* to an area of type *B*. They can do it in two ways: The first method defined by reaction rule *r1* allows each UAV to move separately. The second method, denoted by reaction rule *r2*, allows both UAVs to move in a cooperative manner. One can think of these reaction rules as of different algorithms enabling various capabilities of the UAVs. We do not provide a graphical representation of reaction rules for this example.

We have prepared a software library for generating Tracking Transition Systems available here [38].

Table 1. An example of a Tracking Transition System. Each row defines a single transition in the system. The initial state is defined in the first column of the first row. The definition of two reaction rules used to generate this TTS were omitted but they allow to move either one or two nodes of type U from A to B at once.

Agt	Lab	Agt	Par	Res
	*r1*		$Par = \{0, 1, 3\}$	$\sigma = \{(0,0), (1,2), (2,3), (3,1)\}$
	*r1*		$Par = \{0, 2, 3\}$	$\sigma = \{(0,0), (1,1), (2,3), (3,2)\}$
	*r2*		$Par = \{0, 1, 2, 3\}$	$\sigma = \{(0,0), (1,3), (2,1), (3,2)\}$
	*r1*		$Par = \{0, 1, 2\}$	$\sigma = \{(0,0), (1,2), (2,3), (3,1)\}$

2.2. State Space

Having a Tracking Transition System we can transform it into a UAV swarm mission state space. A state space can be later used to generate a behavior for elements of the swarm we can control or have an influence on. Such elements will be called *agents*.

We have taken the following assumptions regarding modeled systems.

1. The number of agents is constant during whole mission.
2. A system cannot change its state without an explicit action of an agent (alone or in cooperation with other agents).
3. No actions performed by agents is subject to uncertainty.
4. A swarm mission can end for each agent separately in different moment. In other words, agents do not have to finish their part of the mission all at the same time.
5. In case of cooperative actions (actions performed by multiple agents), it is required of all participants to start cooperation at the same moment.

A state space S for a system consisting of n_a agents and n_s states is defined as

$$\mathcal{S} = (N, A, L, I, C, T, M)$$

where

- $N \subset \mathbb{N}$—a set of vertices in the state space. It corresponds to bigraphs in Tracking Transition System;
- $A \subseteq N \times N$—a multiset of ordered pairs of vertices. Called set of directed edges;
- L—a set of labels of changes in the system. It will usually consists of reaction rules names from the Tracking Transition System the state space originate from. To determine what changes, in what order, have led to to a specific state we will additionally introduce a set $R = \{l_t | l \in L, t \in \mathbb{N}\}$.
- $I = \{\mathbb{N}_1^2 \times \cdots \times \mathbb{N}_{n_a}^2\}$—a set of possible state-at-time (SAT) configurations. For example, for $n_a = 2$ the element $i_1 = \langle(0, 777), (1, 123)\rangle$ denotes a situation where the agent

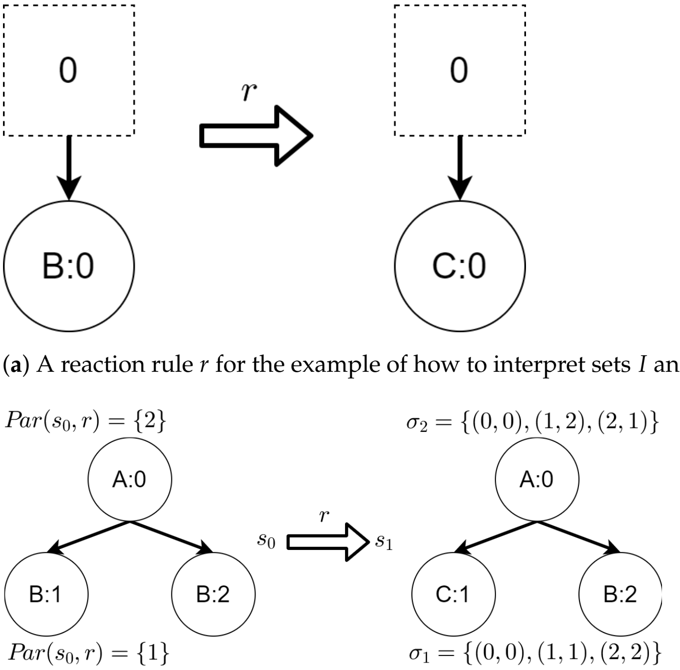
with id 0 is at the moment 777 while the agent with id 1 is at the moment 123. It is important to emphasize that the configuration $i_2 = \langle(1, 123), (0, 777)\rangle$ has the same time interpretation but different spatial interpretation. We later show an example with a justification why we need such a set.

- $C = (I \times 2^R) \cup \{0\}$—a set of possible mission courses. 0 denotes the neutral element, i.e., $\forall_{x \in C} x + 0 = 0 + x = x$, we do not define operation + for the rest of elements of the C set.
- $T = \{f_i : C \times \mathbb{N} \to C | i \in \mathbb{N}\} \cup \{f_{null}\}$—a set of functions defining progress of a mission. We later give an example with a rationale why we need such a set. The f_{null} function returns 0 regardless of input. Additionally, we will denote by $T_{i,j} \subset T$ a set of all mission progress functions from the i state to the j state;
- $M : A \to T$—a bijection mapping of edges to mission progress functions.

Below, we present an example demonstrating why we needed both I and T sets.

Let us assume that some TBRS consists of two bigraphs s_0 and s_1 as in Figure 3b. Reaction rule for this TBRS is presented in Figure 3a, agents in this system are denoted by the control of type B. Then, transform the TBRS into TTS. This TTS consists of two states (associated to both bigraphs) and two transitions (there are two nodes of type B and as we can change only one of them there are two ways to do so). Depending on whether the vertex with id 1 or 2 (numbering according to left-hand side of Figure 3b participates in the reaction the result state-at-time configuration will differ. Let us assume that the SAT configuration for the state associated with bigraph s_0 is equal to $i_0 = \langle(1, 0), (2, 0)\rangle$ and the reaction with label r takes t_r units of time. Depending on which vertex participates in the reaction, the SAT configuration for the state s_1 is either $i_1 = \langle(1, t_r), (2, 0)\rangle$ or $i_1^{\prime} = \langle(2, t_r), (1, 0)\rangle$. Because of this, the corresponding mission progress functions will be of the form $f_1(\big[\langle(a, x), (b, y)\rangle, \Omega\big], t) = \big[\langle(a, x + t_r), (b, y)\rangle, \Omega \cup \{l^2_{t+1}\}\big]$ and $f_2(\big[\langle(a, x), (b, y)\rangle, \Omega\big], t) = \big[\langle(b, y + t_r), (a, x)\rangle, \Omega \cup \{l^1_{t+1}\}\big]$.

Edge identifiers l^1 and l^2 denote which way have led to the s_1 state, their names are arbitrary.



(**a**) A reaction rule r for the example of how to interpret sets I and T. The τ function is identity

(**b**) An example of reaction after applying the above reaction rule. Please note that while the result bigraph remains the same, the σ function differs depending on which vertex participates in the reaction

Figure 3. A TTS for the example of how to interpret sets I and T.

A micro-example of the state space based on TTS from Table 1 is presented in Figure 4 with the mission progress functions defined in Table 2. The key idea behind generating mission progress functions is as follows. For each bigraph B (either source or outcome in a transition), we treat a subset of V_B (denoting identifiers of agents that we want to determine a behavior policy for) as ordered set. We then compare if the order of agents in the source bigraph has changed in the outcome bigraph. If it did, then we must reflect this change in a tuple being an element of I set. In our micro-example, such change of order is particularly visible in the first two transitions (represented by functions f_1 and f_2). Both the source and outcome bigraphs of these transitions are the same, yet in the first transition, the order of agents (UAVs) has changed while in the second transition it has not. This is due to the residue function of both transitions. In the first transition, the order of UAVs identifiers (here 1 and 2 in the source bigraph and 1 and 3 in the outcome) are switched, that is, the order in the source bigraph is $(1,2)$ and the order in the outcome bigraph is $(3,1)$. Because of that, the order of the input tuple in f_1 is changed from $\langle(a,x),(b,y)\rangle$ to $\langle(b,y+1),(a,x)\rangle$. The incrementation of y variable indicates change of time for an agent with identifier equal to value of b. In the second transition, the order remained the same, that is, $(1,2)$ (in source) and $(1,3)$ (in the outcome of the transition). The second case for all mission progress functions, returning 0, is necessary to properly define a walk in the state space. Its usage will be explained in the next subsection.

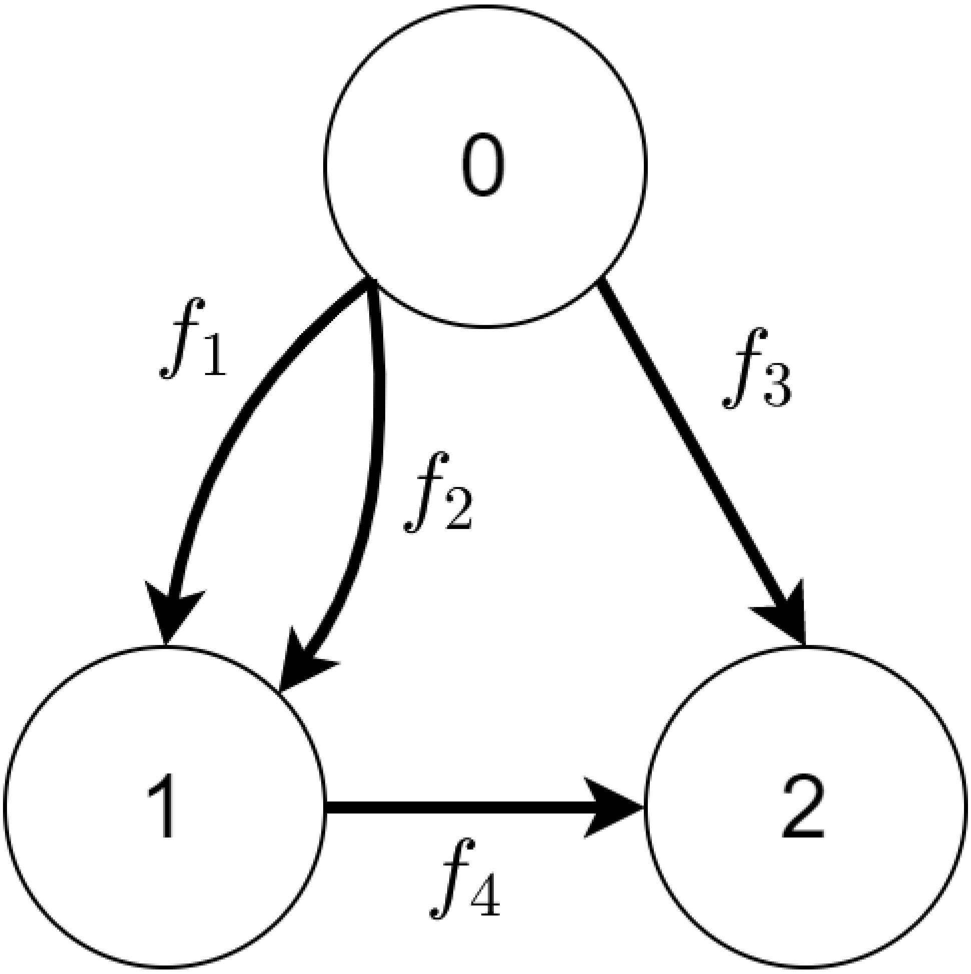


Figure 4. The state space generated from Tracking Transition System defined in Table 1. Mission progress functions definitions are defined in Table 2.

Table 2. Mission progress function definitions for state space presented in Figure 4. All actions defined by reaction rules are assumed to take 1 unit of time. It is worth noting that f_3 function requires both UAVs to be in the same time (variable z) in order to return something other than 0.

Function Identifier	Function Definition
f_1	$f_1(c,t) = \begin{cases} [\langle (b,y),(a,x+1)\rangle, \Omega \cup \{r1_{t+1}\}] & : c = [\langle (a,x),(b,y)\rangle, \Omega] \\ 0 & : c = 0 \end{cases}$
f_2	$f_2(c,t) = \begin{cases} [\langle (a,x),(b,y+1)\rangle, \Omega \cup \{r1_{t+1}\}] & : c = [\langle (a,x),(b,y)\rangle, \Omega] \\ 0 & : c = 0 \end{cases}$
f_3	$f_3(c,t) = \begin{cases} [\langle (a,z+1),(b,z+1)\rangle, \Omega \cup \{r2_{t+1}\}] & : c = [\langle (a,z),(b,z)\rangle, \Omega] \\ 0 & : c \neq [\langle (a,z),(b,z)\rangle, \Omega] \end{cases}$
f_4	$f_4(c,t) = \begin{cases} [\langle (b,y),(a,x+1)\rangle, \Omega \cup \{r1_{t+1}\}] & : c = [\langle (a,x),(b,y)\rangle, \Omega] \\ 0 & : c = 0 \end{cases}$

We omit here an algorithm for transformation of TTS into state space but an exemplary implementation of a software doing it is available at [39].

2.3. Behavior Policy

We define a behavior policy as a schedule of actions for each agent from the beginning of a mission to its end, without breaks.

Having a state space we can view a behavior policy as a walk indicating what changes and done by who are required in order to reach a desired state.

Before we demonstrate how to construct a proper policy behavior based on a state space, we first need to define the following elements. Please note that by a series we will understand a finite sum of elements.

- $K_s^t = c_1 + \cdots + c_m = \sum_{i=1 \cdots m} c_i \quad c_i \in C, s \in \{0, \cdots, n_s - 1\}, t \in \mathbb{N}$—a series, where summands are mission courses leading to the state s;
- $N_K(K_s^t) \in \mathbb{N}$—a function returning a number of elements in a given series. According to the earlier definition, for any series K_s^t this function returns a value of m (the greatest index of c_i);
- $F_{i,j}(x,t) = \sum_{k \in T_{i,j}} f_k(x,t) \quad i,j \in \{0, \cdots, n_s - 1\}, t \in \mathbb{N}$—a series, where summands are mission progress functions from the i to the j state;
- $\mathbb{M}_K^t = \begin{bmatrix} K_0^t & \cdots & K_{n_s-1}^t \end{bmatrix}, t \in \mathbb{N}$—a matrix whose elements are series indicating possible walks leading to each state. Index t denotes a number of steps made in a state space. By a step we understand a transition between vertices (including the situation where the initial and final vertex are the same);
- $\mathbb{M}_F^t = \begin{bmatrix} F_{0,0}(x,t) & \cdots & F_{0,n_s-1}(x,t) \\ \cdots & \cdots & \cdots \\ F_{n_s,0}(x,t) & \cdots & F_{n_s-1,n_s-1}(x,t) \end{bmatrix}$—a matrix of transitions between states.

Furthermore, we define two operations:

- $K_s^t \circ F_{i,j}(x,t) = \sum_{k \in T_{i,j}} \sum_{l=1 \cdots N_K(K_s^t)} f_k(c_l, t)$—a convolution of series defined above;
- $\mathbb{M}_K^{t+1} = \mathbb{M}_K^t \cdot \mathbb{M}_F^t$—a multiplication of matrices defined above. Elements of the new matrix are defined by the formula

$$K_s^{t+1} = \sum_{k=0}^{n_s-1} K_k^t \circ F_{k,s}(x,t)$$

With the elements defined above, we can generate all walks consisting of specified number of steps from the initial state to a final state. To do so, one must define the initial state as a $\mathbb{M}_K^0$ matrix and multiply subsequent results by $\mathbb{M}_F^i$ specified number of times. The result will be a $\mathbb{M}_K^x$ matrix, whose elements in the ith column will contain information about all possible walks with x steps that ends in the ith state of the state space. If an element in the specified column is equal to 0, then it means there is no such walk.

Going back to our micro-example, using the state space as in Figure 4 with function definitions listed in Table 2 we can determine all sequences of actions that lead to the state denoted as 2nd. Each sequence is equivalent to behavior policy that, when applied, results in moving both UAVs to the area of type B.

In order to determine such sequences we create two matrices: a matrix of transitions $\mathbb{M}_F^t$ and matrix of initial state $\mathbb{M}_K^0$. Having both of them, we can multiply subsequent $\mathbb{M}_K^t$ matrices by corresponding $\mathbb{M}_F^t$ matrices and check whether the third state (recall that numbering starts from 0) is reachable. By reachable we understand having value other than 0 in specified column of the $\mathbb{M}_K^t$ matrix.

Definitions of both matrices are listed below.

$$\mathbb{M}_F^t = \begin{bmatrix} f_{null} & f_1 + f_2 & f_3 \\ f_{null} & f_{null} & f_4 \\ f_{null} & f_{null} & f_{null} \end{bmatrix}$$

$$\mathbb{M}_K^0 = \begin{bmatrix} [\langle(1,0),(2,0)\rangle, \varnothing] & 0 & 0 \end{bmatrix}$$

The $\langle(1,0),(2,0)\rangle$ tuple in the first column of $\mathbb{M}_K^0$ matrix denotes that we have two agents. They are identified as agent 1 and agent 2, although this numbering is arbitrary and could be 777 and 111 as well. The zeros in both $(1,0)$ and $(2,0)$ indicate that both agents starts the mission at the same moment.

Subsequent $\mathbb{M}_K^t$ matrices let us determine how system may change when a specified number of actions occur. For example, $\mathbb{M}_K^1$ gives us information how system may evolve when one action occurs and $\mathbb{M}_K^2$ two actions etc.

In this example, $\mathbb{M}_K^1$ and $\mathbb{M}_K^2$ are of the form

$$\mathbb{M}_K^1 = \mathbb{M}_K^0 \cdot \mathbb{M}_F^0 = \begin{bmatrix} [\langle(1,0),(2,0)\rangle, \varnothing] & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_{null}(c,0) & f_1(c,0) + f_2(c,0) & f_3(c,0) \\ f_{null}(c,0) & f_{null}(c,0) & f_4(c,0) \\ f_{null}(c,0) & f_{null}(c,0) & f_{null}(c,0) \end{bmatrix}$$

$$\mathbb{M}_K^1 = \begin{bmatrix} 0 & [\langle(2,0),(1,1)\rangle, \{r1_1\}] + [\langle(1,0),(2,1)\rangle, \{r1_1\}] & [\langle(1,1),(2,1)\rangle, \{r2_1\}] \end{bmatrix}$$

$$\mathbb{M}_K^2 = \mathbb{M}_K^1 \cdot \mathbb{M}_F^1 = \begin{bmatrix} [\langle(1,0),(2,0)\rangle, \varnothing] & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} f_{null}(c,1) & f_1(c,0) + f_2(c,1) & f_3(c,1) \\ f_{null}(c,1) & f_{null}(c,1) & f_4(c,1) \\ f_{null}(c,1) & f_{null}(c,1) & f_{null}(c,1) \end{bmatrix}$$

$$\mathbb{M}_K^2 = \begin{bmatrix} 0 & 0 & [\langle(1,1),(2,1)\rangle, \{r1_1, r1_2\}] + [\langle(2,1),(1,1)\rangle, \{r1_1, r1_2\}] \end{bmatrix}$$

We have prepared a software library for generating behavior policies, available here [40].

3. Results

The result of our work is a method of design robotic swarms (specifically UAV swarms). In this section, we will present this method with a use of two scenarios related to UAV swarms.

3.1. Introductory Scenario

The first of the discussed scenarios aims at presenting how elements defined in Section 2 are meant to be used in modeling a UAV swarm mission. The modeled mission will include the following aspects typical for swarm robotics tasks:

- Cooperation
- Multiple ways of achieving a goal
- Synchronization of agents by "idling" while waiting for others

The scenario consists of a map composed of two types of areas and two UAVs of different kind. Let us name the types of areas X and Y while types of UAVs will be called A and B. The initial state of the mission is presented (as a bigraph) in Figure 5. The goal of the mission is to move both UAVs from their initial location to the area of type Y (there is only one such area). In order to demonstrate cooperation, we will restrict possibility of moving from an area of type X to an area of type Y only to simultaneous transition by both UAVs. This can be interpreted in various ways. For example one kind of the UAVs can serve as a navigator when moving between different types of areas, or it can be a carrier that allows the other UAV to travel longer distances. We will not choose one interpretation over the other and rather focus on how to deal with such situations. To additionally increase complexity of the mission we will assume that different kind of UAVs move at different pace. This will enforce a behavior policy to include actions that only purpose is to "kill time" by one UAV while the other, the slower one, will finish their part before engaging in cooperation. It is important to emphasize that we allow agents (UAVs in this example) to idle only by performing actions (either alone or in cooperation) from a fixed set of actions represented by reaction rules.

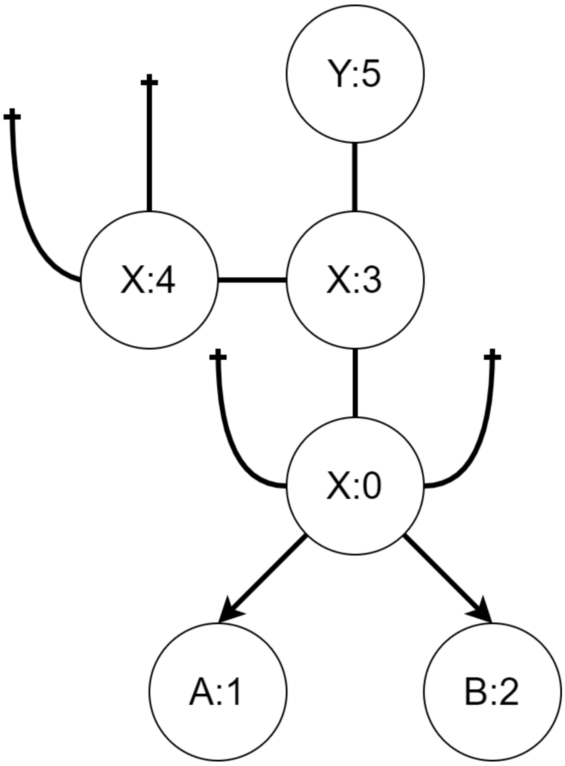
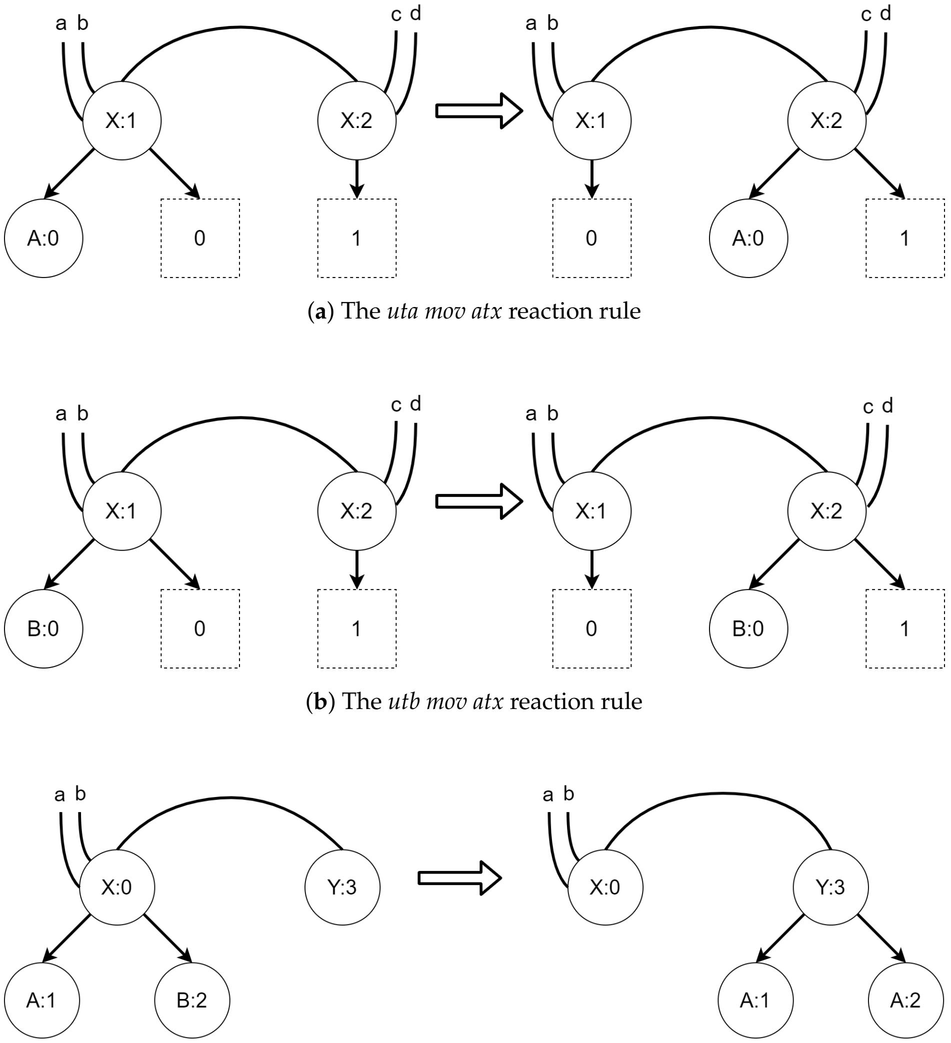


Figure 5. Initial state of the system modeled in scenario 1. X and Y are different types of areas, while A and B are different kinds of UAVs.

3.1.1. Bigraphical Reactive System

The TBRS for the first scenario consists of three reaction rules and six bigraphs. The initial state is shown in Figure 5. Controls *A* and *B* represent UAVs of a different kind, while controls *X* and *Y* denote areas of the type with the same name. The reaction rules for the TBRS of this system are presented in Figure 6. The *uta mov atx* reaction, presented in Figure 6a, allows a UAV of type A to move between areas of type X. The *utb mov atx* reaction, depicted in Figure 6b, allows a UAV of type B to do the same. Finally, the *mov atx2y* rule allows for transition of both UAVs between an area of type X to an area of type Y; it is presented in Figure 6c.



(**a**) The *uta mov atx* reaction rule

(**b**) The *utb mov atx* reaction rule

(**c**) The *mov atx2y* reaction rule

Figure 6. The reaction rules for the first scenario. All of the site mappings and residue functions are identities.

The transitions in Tracking Transition System for this scenario are listed in Appendix A. For simple reference, below is the list of descriptions of each state.

- State 0—both UAVs are in the same area of type X that is not at the center of the map. It does not matter which one since all combinations are isomorphic to each other.
- State 1—the UAV of type A is at the center while the UAV of type B is in an area of type X.
- State 2—the UAV of type B is at the center while the UAV of type A is in an area of type X.
- State 3—each of the UAVs is in a distinct area of type X that is not at the center of the map.
- State 4—both of the UAVs are at the center of the map.
- State 5—both of the UAVs are in the area of type Y.

3.1.2. State Space

The state space generated from the TTS described above and defined in Appendix A is presented in Figure 7. The set of labels of changes was defined as

$$L = \{m1^{1}, m2^{2}, m1^{3}, m2^{4}, m1^{5}, m2^{6}, m1^{7}, m2^{8}, m2^{9}, m1^{10}, m3^{11}, m1^{12}, m1^{13}, m2^{14}, m2^{15}\}$$

Mapping of labels to reaction rules was listed in Table 3. For every transition in the TTS there is a corresponding edge in the state space.

As it was stated in the introduction, each type of UAVs moves with different speed. We took the assumption that the UAV of type A needs 1 unit of time to move between areas of type X and the UAV of type B needs 3 units of type to do the same job. Moving between an area of type X to an area of type Y takes 4 units of time (it is done only by two UAVs at once so there is no differentiation by types).

Functions assigned to edges are defined below.

$$f_1(c,t) = \begin{cases} \big[\langle (a,x+1),(b,y)\rangle, \Omega \cup \{m1^{1}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_2(c,t) = \begin{cases} \big[\langle (b,y+3),(a,x)\rangle, \Omega \cup \{m2^{2}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_3(c,t) = \begin{cases} \big[\langle (a,x+1),(b,y)\rangle, \Omega \cup \{m1^{3}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_4(c,t) = \begin{cases} \big[\langle (b,y+3),(a,x)\rangle, \Omega \cup \{m2^{4}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_5(c,t) = \begin{cases} \big[\langle (a,x+1),(b,y)\rangle, \Omega \cup \{m1^{5}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_6(c,t) = \begin{cases} \big[\langle (b,y),(a,x+3)\rangle, \Omega \cup \{m2^{6}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_7(c,t) = \begin{cases} \big[\langle (a,x),(b,y+1)\rangle, \Omega \cup \{m1^{7}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_8(c,t) = \begin{cases} \big[\langle (b,y),(a,x+3)\rangle, \Omega \cup \{m2^{8}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_9(c,t) = \begin{cases} \big[\langle (b,y+3),(a,x)\rangle, \Omega \cup \{m2^{9}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_{10}(c,t) = \begin{cases} \big[\langle (a,x+1),(b,y)\rangle, \Omega \cup \{m1^{10}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_{11}(c,t) = \begin{cases} \big[\langle (b,z+4),(a,z+4)\rangle, \Omega \cup \{m3^{11}_{t+1}\}\big] & : \ c = [\langle (a,z),(b,z)\rangle, \Omega] \\ 0 & : \ c \neq [\langle (a,z),(b,z)\rangle, \Omega] \end{cases}$$

$$f_{12}(c,t) = \begin{cases} \big[\langle (a,x),(b,y+1)\rangle, \Omega \cup \{m1^{12}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_{13}(c,t) = \begin{cases} \big[\langle (a,x),(b,y+1)\rangle, \Omega \cup \{m1^{13}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_{14}(c,t) = \begin{cases} \big[\langle (b,y),(a,x+3)\rangle, \Omega \cup \{m2^{14}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

$$f_{15}(c,t) = \begin{cases} \big[\langle (b,y),(a,x+3)\rangle, \Omega \cup \{m2^{15}_{t+1}\}\big] & : \ c = \big[\langle (a,x),(b,y)\rangle, \Omega\big] \\ 0 & : \ c = 0 \end{cases}$$

Table 3. Mapping of labels in the state space to reaction rules in the TTS for scenario 1.

Label of Change	Reaction Rule
$m1^i$	*uta mov atx* occurring in *i*th transition
$m2^i$	*utb mov atx* occurring in *i*th transition
$m3^i$	*mov atx2y* occurring in *i*th transition

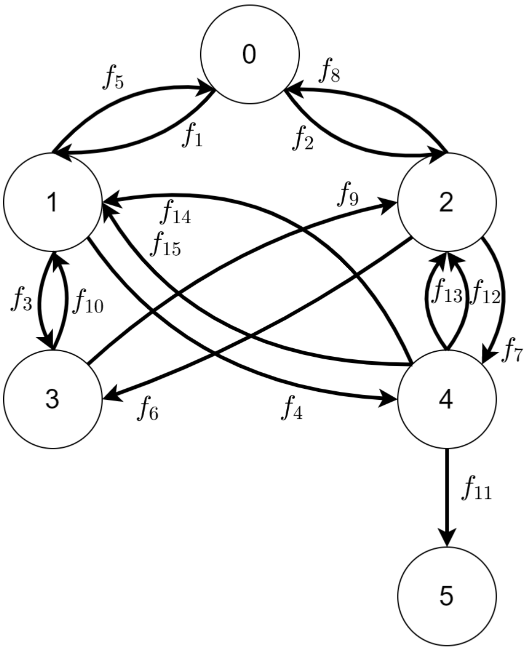


Figure 7. The state space for scenario 1 based on Tracking Transition System defined in Appendix A.

3.1.3. Behavioral Policy

The initial state for scenario 1 is represented by the vertex with id 0 in Figure 7, the final state is the one with id 5. Our goal is to find a shortest walk from vertex with id 0 to the vertex with id 5; it must not violate the constraint that allow agents to cooperate only when they start in the same moment of time.

Each $T_{i,j}$ set is defined in Table A2 of Appendix B.

Based on T set, we can construct $\mathbb{M}_F^t$:

$$\mathbb{M}_F^t = \begin{bmatrix} f_{null} & f_1 & f_2 & f_{null} & f_{null} & f_{null} \\ f_5 & f_{null} & f_{null} & f_3 & f_4 & f_{null} \\ f_8 & f_{null} & f_{null} & f_6 & f_7 & f_{null} \\ f_{null} & f_{10} & f_9 & f_{null} & f_{null} & f_{null} \\ f_{null} & f_{14}+f_{15} & f_{12}+f_{13} & f_{null} & f_{null} & f_{11} \\ f_{null} & f_{null} & f_{null} & f_{null} & f_{null} & f_{null} \end{bmatrix}$$

Knowing that the initial state is associated with vertex with id 0, the $\mathbb{M}_K^0$ is of the form

$$\mathbb{M}_K^0 = \begin{bmatrix} [\langle(1,0),(2,0)\rangle, \varnothing] & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

By multiplying the above matrices we gain information about changes of the system and how will it affect the agents (UAVs in this case).

For example:

$$\mathbb{M}_K^1 = \mathbb{M}_K^0 \cdot \mathbb{M}_F^0 = \begin{bmatrix} 0 & [\langle(1,1),(2,0)\rangle, \{m1_1\}] & [\langle(2,3),(1,0)\rangle, \{m2_1\}] & 0 & 0 & 0 \end{bmatrix}$$

In turn, the $\mathbb{M}_K^5$ matrix gives us information about every possible walk leading to the state with id 5 (in general, it gives information about all possible walks consisting of 5 edges). Below are listed all elements of the series being the element of the 6th column (numbering from 0) in the $\mathbb{M}_K^5$ matrix:

- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m2_4^4, m1_3^1, m1_2^5, m1_1^1\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m2_4^4, m1_3^1, m1_2^3, m1_1^1\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m2_3^2, m1_2^5, m1_1^1\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m2_3^9, m1_2^3, m1_1^1\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m2_4^7, m1_3^{13}, m2_2^4, m1_1^1\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m1_3^{13}, m1_2^7, m2_1^2\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m1_3^{12}, m2_2^4, m1_1^1\}\big]$
- $\big[\langle(1,7),(2,7)\rangle, \{m3_5^{11}, m1_4^7, m1_3^{12}, m1_2^7, m2_1^2\}\big]$

Using the above we can define a behavior policy (i.e., a schedule of actions) for each UAV.

For example using the first element of the 6th column we get the following walk.

$$0 \xrightarrow{m1_1^1} 1 \xrightarrow{m1_2^5} 0 \xrightarrow{m1_3^1} 1 \xrightarrow{m2_4^4} 4 \xrightarrow{m3_5^{11}} 5$$

This can be further transformed into a behavior policy as presented in Table 4.

Table 4. A schedule of actions for both UAVs based on the walk of the form $0 \xrightarrow{m1_1^1} 1 \xrightarrow{m1_2^5} 0 \xrightarrow{m1_3^1} 1 \xrightarrow{m2_4^4} 4 \xrightarrow{m3_5^{11}} 5$. Each (x, y) element denotes: x—last scheduled action, y—the time moment since the x action is performed.

	START	$\boldsymbol{m1_1^1}$	$\boldsymbol{m1_2^5}$	$\boldsymbol{m1_3^1}$	$\boldsymbol{m2_4^4}$	$\boldsymbol{m3_5^{11}}$	**END**
UAV_A	$(-,0)$	$(m1,0)$	$(m1,1)$	$(m1,2)$	$(m1,2)$	$(m3,3)$	$(m3,3)$
UAV_B	$(-,0)$	$(-,0)$	$(-,0)$	$(-,0)$	$(m2,0)$	$(m3,3)$	$(m3,3)$

3.2. More Advanced Example

The second example is intended to present a more realistic UAV swarm mission. Additionally, we present two propositions of metrics for measuring size of swarm robotic systems.

In this scenario, the goal is to collect all information located on a map and secure it in a base. We have made the following assumptions, in regard to the mission.

- Every UAV is capable of storing and transporting up to one information at the time.
- All sources of information can transmit information to any number of UAVs in parallel.
- An information can be secured in a base only when the UAV containing the information is inside the base.
- Any number of UAVs can secure information in a base at the same time.

The above mission will be resolved in four different variants of the initial state:

1. The map consists of two areas, one UAV, one source of information, and one information.
2. The map consists of four areas, two UAVs, two sources of information, and two information.
3. The map consists of four areas, two UAVs, two sources of information, and four information.
4. The map consists of nine areas, three UAVs, two sources of information, and four information.

3.2.1. Bigraphical Reactive System

Regardless of the variant, the BRS for the second scenario consists of six reaction rules, presented in Figures 8–13. All of the initial states for different variants of the scenario are presented in Figure 14. All site mapping functions and residue functions are identities. Table 5 lists all control types with respective real world objects they are representing.

1. the *move* reaction rule:

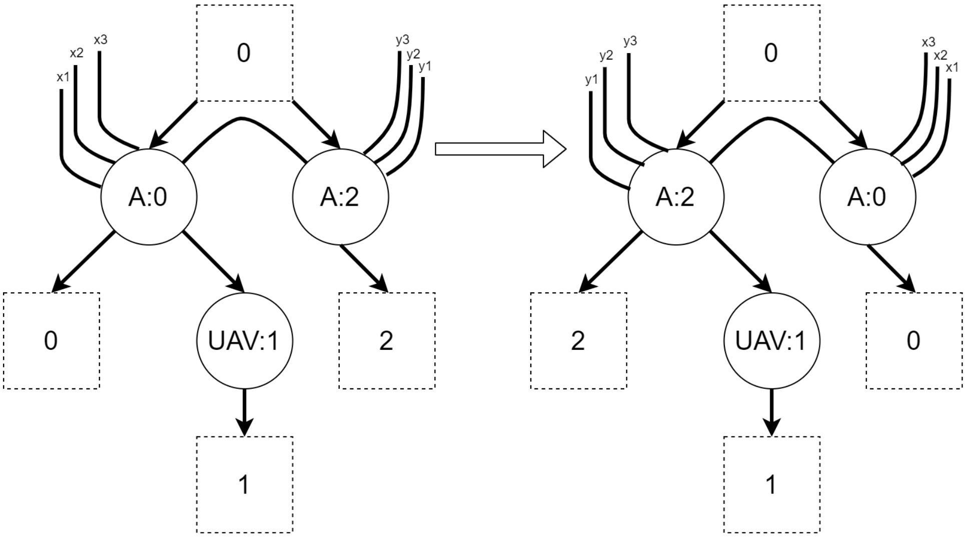


Figure 8. Reaction rule *move* for scenario 2.

Rationale: the rule is intended to allow a UAV to move between areas.

2. The *move into base* reaction rule:

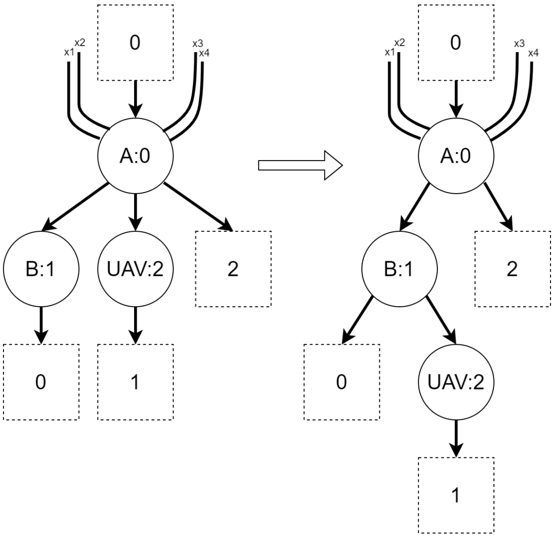


Figure 9. Reaction rule *move into base* for scenario 2.

Rationale: the rule is intended to allow a UAV to move into a base.

3. The *move out of base* reaction rule:

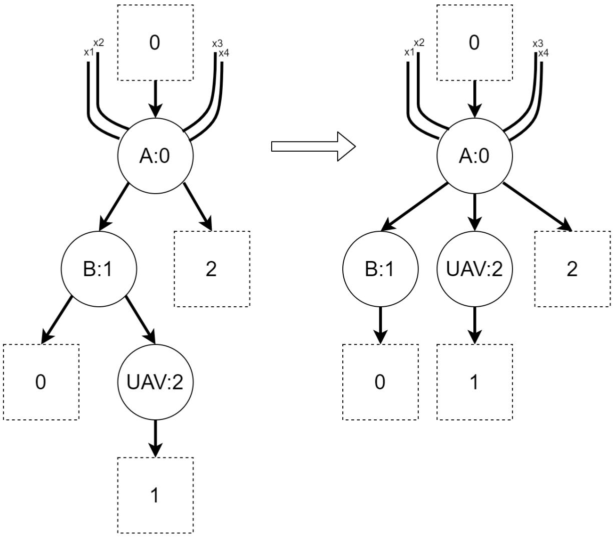


Figure 10. Reaction rule *move out of base* for scenario 2.

Rationale: the rule allows to move a UAV out of base.

4. The *download data* reaction rule:

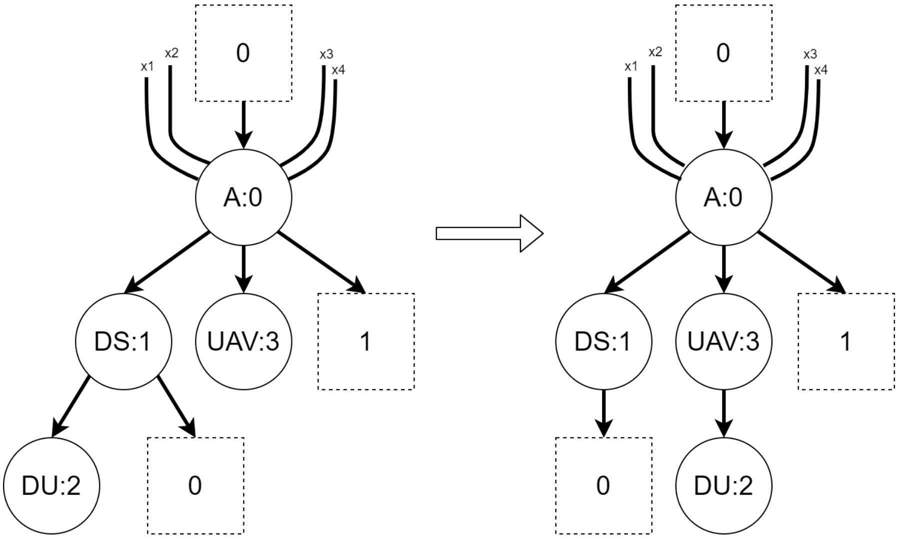


Figure 11. Reaction rule *download data* for scenario 2.

Rationale: the rule is intended to allow a UAV to download information from an information source.

5. The *deploy data* reaction rule:

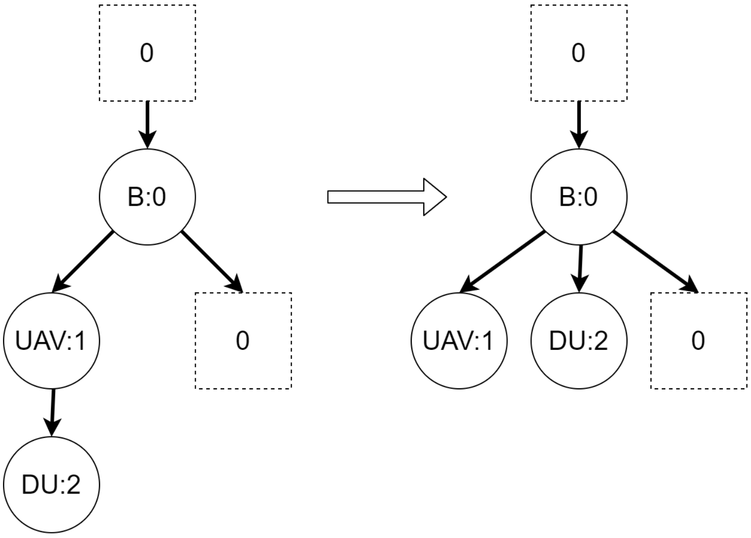


Figure 12. Reaction rule *deploy data* for scenario 2.

Rationale: the rule is intended to allow UAVs to secure information in a base.

6. The *deactivate uav* reaction rule:

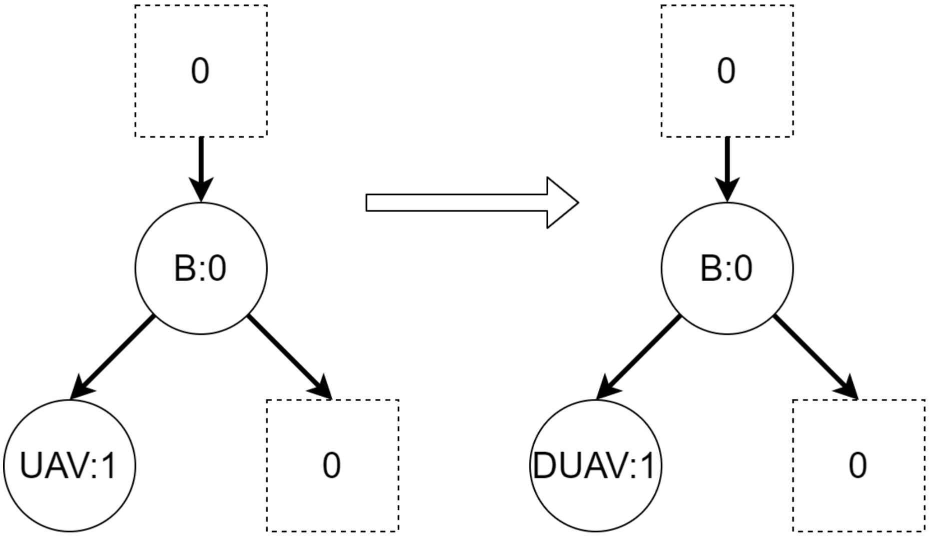


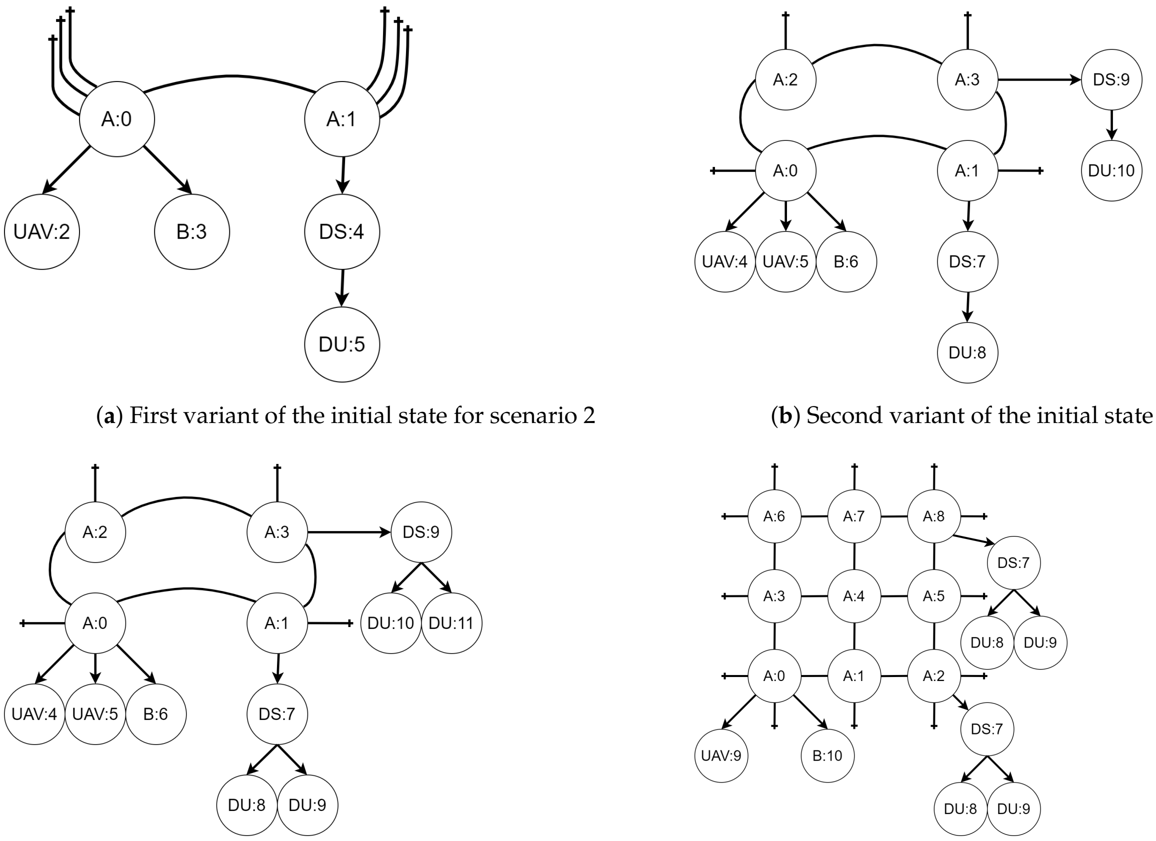
Figure 13. Reaction rule *deactivate uav* for scenario 2.

Rationale: the rule is intended to allow a UAV to get deactivated. We will consider the mission be finished when all information are secured and all UAVs are deactivated.

Table 5. A list of controls with their respective interpretations.

Control	Interpretation
A	A map area
UAV	An active Unmanned Aerial Vehicle
DUAV	An inactive Unmanned Aerial Vehicle
B	The base
DS	A source of information
DU	A unit (piece) of information

Initial states of the system in each variant are presented in Figure 14a–d. The numbering corresponds to the sequence of the variants descriptions above.



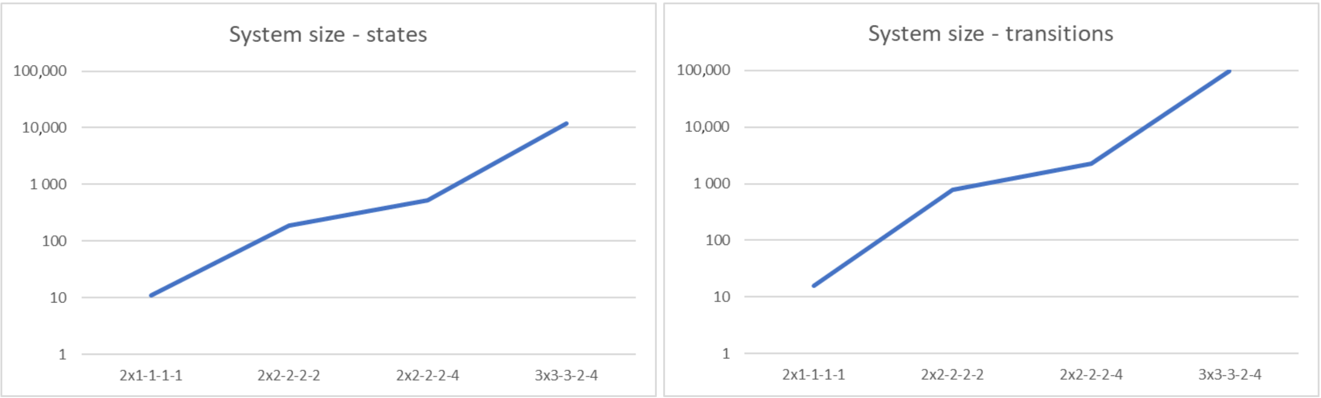
(**a**) First variant of the initial state for scenario 2 (**b**) Second variant of the initial state for scenario 2

(**c**) Third variant of the initial state for scenario 2 (**d**) Fourth variant of the initial state for scenario 2

Figure 14. Initial states of scenario 2 in all variants.

3.2.2. State Space

From the above reactive systems (each generated from different initial state), we can generate four state spaces. Due to their large size (see Figure 15), their details, such as graphical representation and transition matrices, will be defined only for the first (smallest) variant.



(**a**) Number of states in respect to the variant of scenario 2 (**b**) Number of transitions in respect to the variant of scenario 2

Figure 15. Size of scenario 2 system for all its variants.

A relation between variant of scenario 2 and its size is depicted in Figure 15. The state space generated from TTS of the first variant of scenario 2 is shown in Figure 16.

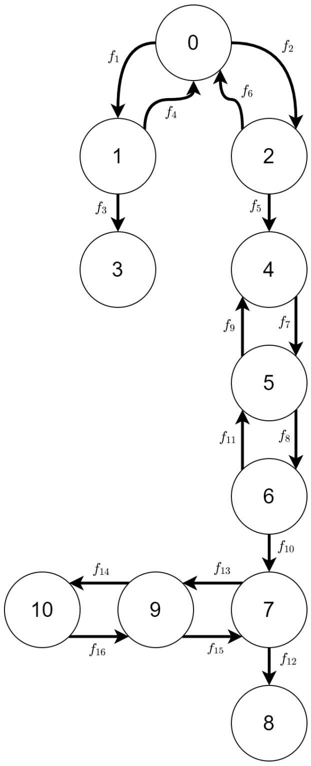


Figure 16. The state space of the first variant of scenario 2.

Mission progress functions definitions are listed below.

$$f_1([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mib_{t+1}\}]$$

$$f_2([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mov_{t+1}\}]$$

$$f_3([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{dea_{t+1}\}]$$

$$f_4([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mob_{t+1}\}]$$

$$f_5([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{dod_{t+1}\}]$$

$$f_6([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mov_{t+1}\}]$$

$$f_7([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mov_{t+1}\}]$$

$$f_8([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mib_{t+1}\}]$$

$$f_9([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mov_{t+1}\}]$$

$$f_{10}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{ded_{t+1}\}]$$

$$f_{11}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mob_{t+1}\}]$$

$$f_{12}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{dea_{t+1}\}]$$

$$f_{13}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mob_{t+1}\}]$$

$$f_{14}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mov_{t+1}\}]$$

$$f_{15}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mib_{t+1}\}]$$

$$f_{16}([\langle(a,x)\rangle,\Omega],t) = [\langle(a,x+1)\rangle,\Omega \cup \{mov_{t+1}\}]$$

Table 6 shows the mapping of labels of changes in the system to reaction rules that led to the changes between states.

Table 6. A mapping of system changes labels to corresponding reaction rules.

Label of A Change in A System	Corresponding Reaction Rule
mov	move
mib	move into base
mob	move out of base
ded	deploy data
dod	download data
dea	deactivate uav

3.2.3. Behavior Policy

Similarly as in Section 3.2.2, we will limit ourselves to the first variant of scenario 2. The $T_{i,j}$ sets for the first variant of scenario 2 are listed in Table A3 in Appendix B.

Assuming, the initial state is represented by the vertex with id 0, the $\mathbb{M}_K^0$ matrix is of the form

$$\mathbb{M}_K^0 = \big[[\langle(1,0)\rangle, \varnothing]0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\big]$$

We will omit the definition of $\mathbb{M}_F^t$, hoping that its form is obvious knowing the sets presented in Appendix B.

If the final state is represented by the vertex with id 8, then the first walk will be found in $\mathbb{M}_K^6$ matrix and it will be of the form

$$0 \xrightarrow{mov_1} 2 \xrightarrow{dod_2} 4 \xrightarrow{mov_3} 5 \xrightarrow{mib_4} 6 \xrightarrow{ded_5} 7 \xrightarrow{dea_6} 8$$

4. Discussion

In this paper, we have presented a method of modeling a UAV swarm mission using bigraphs with tracking as well as a method of generating a behavior policy for elements of the swarm. The proposed method has the desired properties described in the introduction. One of the main advantages of the proposed method is the possibility to fully automate the process of determining the behavior of swarm elements from the phase of defining mission requirements (as bigraphical patterns) and UAV capabilities. The proposed method is flexible in terms of using it for different swarm tasks; it is also modular and capable of generating behavior policies on multiple levels of abstractions. Because of its modularity, we can modify some of the method's modules while leaving the rest unchanged. For example, there may be a need of defining a function to evaluate walks generated in the last stage of a design process. It is possible without modifying neither the way the mission is modeled (the previous step) nor how the schedule of actions is constructed based on the result walk (the next step). The method was verified on two scenarios. Additionally, a software to verify the calculated results has been developed.

One of the conclusions from the work is the observation of how quickly a system's size is increasing. In case of a real-world scenario, we can safely assume a size of the system to be in the order of millions of states and dozens of millions of transitions. Based on our tests, current software is capable to effectively support a designer in a process of modeling a system consisting of up to dozens of thousands of states. Because of this, it is reasonable to point out that in order to use our method for a real world use case it is needed to develop more efficient implementations of operations on bigraphs and operations defined in this article. To the best of our knowledge, none of the existing methods of design of UAV swarms are universal, they are either suitable for a specific task or a single group of them (such as *spatial organization* or *decision-making*) at best. Despite classifying our method as problem-agnostic we are aware that assuming that system's change can only be triggered by its agents and non-adaptiveness of behavior policy are quite restrictive. This is a limitation of its applicability to tasks from *spatial organization* or *collective motion* category. Some of the tasks in the latter category may regard situations where environment can significantly change regardless of UAVs actions, an example of this may be a search and rescue operation considered as a special case of foraging task. In such cases, our method is not suitable in its current form. Summarizing, we can recommend our approach to tasks in *spatial organization* category in general and a subset of *collective motion* tasks. *Decision-making* tasks are generally unsuitable to be solved by our method because the behavior policy it generates is non-adaptive and most of the decisions in this category concern events occurring at hand.

Currently, we are unable to quantitatively compare our method to any alternatives. The key factor that may determine whether to choose our approach over others may be the need of reusability. The method presented in this paper generates a behavior policy for a specific task and is obviously inferior in situations where there is a need for generic behavior that is adjustable with some parameters.

We can distinguish the following goals for future work.

- Developing a method of policy behavior generation that takes into account agents that are beyond control of the designer. An example of such agent might be a person.
- Developing a method of behavior policy generation that considers indeterministic nature of agents' actions.
- Developing a method of behavior policy generation for missions with a variable number of agents over their course.
- More efficient software providing the functionality described in this paper. The current version of the software should be considered as proof-of-concept.

Author Contributions: Conceptualization, P.C. and Z.Z.; methodology, P.C. and Z.Z.; software, P.C.; validation, P.C.; formal analysis, P.C.; investigation, P.C.; resources, P.C.; data curation, P.C.; writing—original draft preparation, P.C.; writing—review and editing, Z.Z.; visualization, P.C.; supervision, Z.Z.; project administration, Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. The Tracking Transition System for scenario 1. Each row defines a single transition in the system.

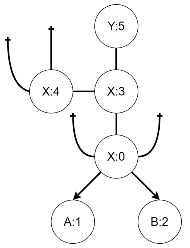
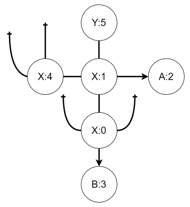
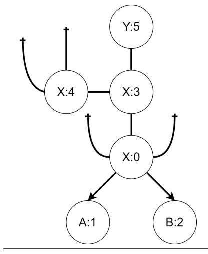
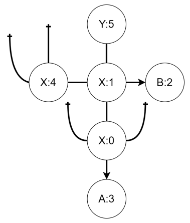
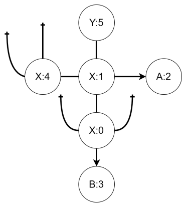
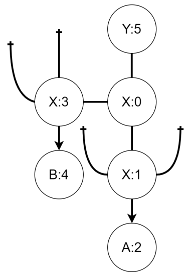
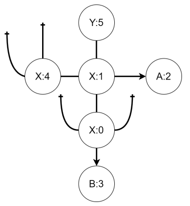
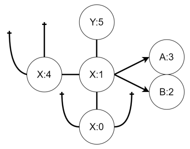
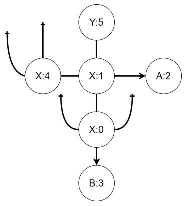
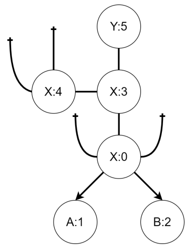
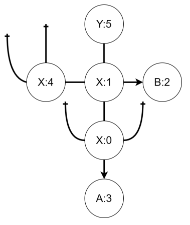
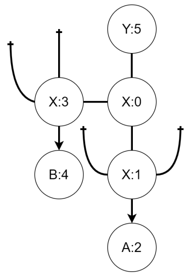
Agt	Lab	Agt	Par	Res
	*uta mov atx*		$\{0,1,3\}$	$\{(0,0),(1,3),$ $(2,1),(3,2),$ $(4,4),(5,5)\}$
	*utb mov atx*		$\{0,2,3\}$	$\{(0,0),(1,3),$ $(2,2),(3,1),$ $(4,4),(5,5)\}$
	*uta mov atx*		$\{1,2,4\}$	$\{(0,1),(1,4),$ $(2,2),(3,0),$ $(4,3),(5,5)\}$
	*atb mov atx*		$\{0,3,1\}$	$\{(0,4),(1,1),$ $(2,3),(3,2),$ $(4,0),(5,5)\}$
	*uta mov atx*		$\{1,2,0\}$	$\{(0,0),(1,2),$ $(2,3),(3,1),$ $(4,4),(5,5)\}$
	*utb mov atx*		$\{1,2,4\}$	$\{(0,1),(1,0),$ $(2,3),(3,4),$ $(4,2),(5,5)\}$

Table A1. *Cont.*

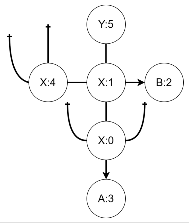
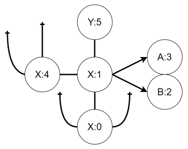
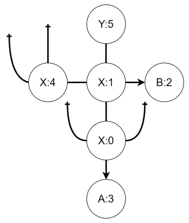
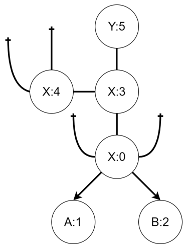
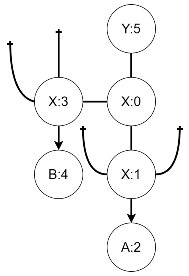
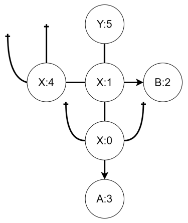
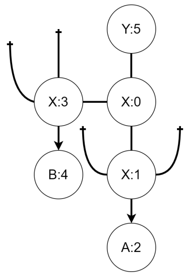
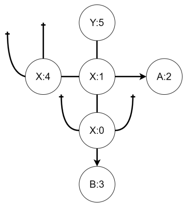
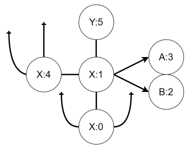
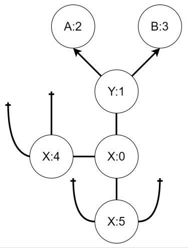
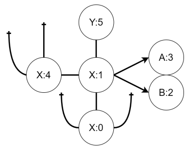
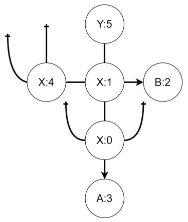
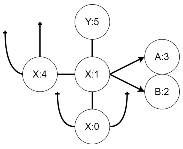
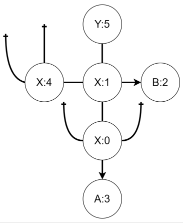
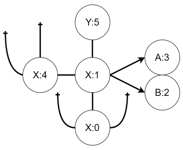
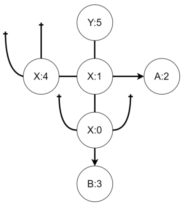
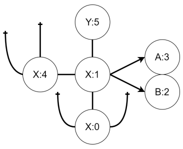
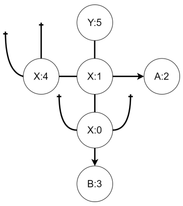
Agt	Lab	Agt	Par	Res
	*uta mov atx*		$\{0,3,1\}$	$\{(0,4),(1,1),$ $(2,2),(3,3),$ $(4,0),(5,5)\}$
	*utb mov atx*		$\{1,2,0\}$	$\{(0,0),(1,3),$ $(2,2),(3,1),$ $(4,4),(5,5)\}$
	*utb mov atx*		$\{3,4,0\}$	$\{(0,1),(1,0),$ $(2,4),(3,2),$ $(4,3),(5,5)\}$
	*uta mov atx*		$\{1,2,0\}$	$\{(0,3),(1,0),$ $(2,2),(3,4),$ $(4,1),(5,5)\}$
	*mov atx2y*		$\{1,3,2,5\}$	$\{(0,1),(1,5),$ $(2,3),(3,2),$ $(4,4),(5,0)\}$
	*uta mov atx*		$\{1,3,4\}$	$\{(0,4),(1,1),$ $(2,2),(3,3),$ $(4,0),(5,5)\}$

Table A1. *Cont.*

Agt	Lab	Agt	Par	Res
	*uta mov atx*		$\{1,3,0\}$	$\{(0,0),(1,1),$ $(2,2),(3,3),$ $(4,4),(5,5)\}$
	*utb mov atx*		$\{1,2,0\}$	$\{(0,0),(1,1),$ $(2,3),(3,2),$ $(4,4),(5,5)\}$
	*utb mov atx*		$\{1,2,4\}$	$\{(0,4),(1,1),$ $(2,3),(3,2),$ $(4,0),(5,5)\}$

Appendix B

Table A2. Elements of the *T* set for scenario 1.

$T_{0,0} = \{f_{null}\}$	$T_{0,1} = \{f_1\}$	$T_{0,2} = \{f_{f_2}\}$	$T_{0,3} = \{f_{null}\}$	$T_{0,4} = \{f_{null}\}$	$T_{0,5} = \{f_{null}\}$
$T_{1,0} = \{f_5\}$	$T_{1,1} = \{f_{null}\}$	$T_{1,2} = \{f_{null}\}$	$T_{1,3} = \{f_3\}$	$T_{1,4} = \{f_4\}$	$T_{1,5} = \{f_{null}\}$
$T_{2,0} = \{f_8\}$	$T_{2,1} = \{f_{null}\}$	$T_{2,2} = \{f_{null}\}$	$T_{2,3} = \{f_6\}$	$T_{2,4} = \{f_7\}$	$T_{2,5} = \{f_{null}\}$
$T_{3,0} = \{f_{null}\}$	$T_{3,1} = \{f_{10}\}$	$T_{3,2} = \{f_9\}$	$T_{3,3} = \{f_{null}\}$	$T_{3,4} = \{f_{null}\}$	$T_{3,5} = \{f_{null}\}$
$T_{4,0} = \{f_{null}\}$	$T_{4,1} = \{f_{14}, f_{15}\}$	$T_{4,2} = \{f_{12}, f_{13}\}$	$T_{4,3} = \{f_{null}\}$	$T_{4,4} = \{f_{null}\}$	$T_{4,5} = \{f_{11}\}$
$T_{5,0} = \{f_{null}\}$	$T_{5,1} = \{f_{null}\}$	$T_{5,2} = \{f_{null}\}$	$T_{5,3} = \{f_{null}\}$	$T_{5,4} = \{f_{null}\}$	$T_{5,5} = \{f_{null}\}$

Table A3. Elements of the T set for scenario 2.

$T_{0,0} = \{f_n\}$	$T_{0,1} = \{f_1\}$	$T_{0,2} = \{f_2\}$	$T_{0,3} = \{f_n\}$	$T_{0,4} = \{f_n\}$	$T_{0,5} = \{f_n\}$	$T_{0,6} = \{f_n\}$	$T_{0,7} = \{f_n\}$	$T_{0,8} = \{f_n\}$	$T_{0,9} = \{f_n\}$	$T_{0,10} = \{f_n\}$
$T_{1,0} = \{f_4\}$	$T_{1,1} = \{f_n\}$	$T_{1,2} = \{f_n\}$	$T_{1,3} = \{f_3\}$	$T_{1,4} = \{f_n\}$	$T_{1,5} = \{f_n\}$	$T_{1,6} = \{f_n\}$	$T_{1,7} = \{f_n\}$	$T_{1,8} = \{f_n\}$	$T_{1,9} = \{f_n\}$	$T_{1,10} = \{f_n\}$
$T_{2,0} = \{f_6\}$	$T_{2,1} = \{f_n\}$	$T_{2,2} = \{f_n\}$	$T_{2,3} = \{f_n\}$	$T_{2,4} = \{f_5\}$	$T_{2,5} = \{f_n\}$	$T_{2,6} = \{f_n\}$	$T_{2,7} = \{f_n\}$	$T_{2,8} = \{f_n\}$	$T_{2,9} = \{f_n\}$	$T_{2,10} = \{f_n\}$
$T_{3,0} = \{f_n\}$	$T_{3,1} = \{f_n\}$	$T_{3,2} = \{f_n\}$	$T_{3,3} = \{f_n\}$	$T_{3,4} = \{f_n\}$	$T_{3,5} = \{f_n\}$	$T_{3,6} = \{f_n\}$	$T_{3,7} = \{f_n\}$	$T_{3,8} = \{f_n\}$	$T_{3,9} = \{f_n\}$	$T_{3,10} = \{f_n\}$
$T_{4,0} = \{f_n\}$	$T_{4,1} = \{f_n\}$	$T_{4,2} = \{f_n\}$	$T_{4,3} = \{f_n\}$	$T_{4,4} = \{f_n\}$	$T_{4,5} = \{f_7\}$	$T_{4,6} = \{f_n\}$	$T_{4,7} = \{f_n\}$	$T_{4,8} = \{f_n\}$	$T_{4,9} = \{f_n\}$	$T_{4,10} = \{f_n\}$
$T_{5,0} = \{f_n\}$	$T_{5,1} = \{f_n\}$	$T_{5,2} = \{f_n\}$	$T_{5,3} = \{f_n\}$	$T_{5,4} = \{f_9\}$	$T_{5,5} = \{f_n\}$	$T_{5,6} = \{f_8\}$	$T_{5,7} = \{f_n\}$	$T_{5,8} = \{f_n\}$	$T_{5,9} = \{f_n\}$	$T_{5,10} = \{f_n\}$
$T_{6,0} = \{f_n\}$	$T_{6,1} = \{f_n\}$	$T_{6,2} = \{f_n\}$	$T_{6,3} = \{f_n\}$	$T_{6,4} = \{f_n\}$	$T_{6,5} = \{f_{11}\}$	$T_{6,6} = \{f_n\}$	$T_{6,7} = \{f_{10}\}$	$T_{6,8} = \{f_n\}$	$T_{6,9} = \{f_n\}$	$T_{6,10} = \{f_n\}$
$T_{7,0} = \{f_n\}$	$T_{7,1} = \{f_n\}$	$T_{7,2} = \{f_n\}$	$T_{7,3} = \{f_n\}$	$T_{7,4} = \{f_n\}$	$T_{7,5} = \{f_n\}$	$T_{7,6} = \{f_n\}$	$T_{7,7} = \{f_n\}$	$T_{7,8} = \{f_{12}\}$	$T_{7,9} = \{f_{13}\}$	$T_{7,10} = \{f_n\}$
$T_{8,0} = \{f_n\}$	$T_{8,1} = \{f_n\}$	$T_{8,2} = \{f_n\}$	$T_{8,3} = \{f_n\}$	$T_{8,4} = \{f_n\}$	$T_{8,5} = \{f_n\}$	$T_{8,6} = \{f_n\}$	$T_{8,7} = \{f_n\}$	$T_{8,8} = \{f_n\}$	$T_{8,9} = \{f_n\}$	$T_{8,10} = \{f_n\}$
$T_{9,0} = \{f_n\}$	$T_{9,1} = \{f_n\}$	$T_{9,2} = \{f_n\}$	$T_{9,3} = \{f_n\}$	$T_{9,4} = \{f_n\}$	$T_{9,5} = \{f_n\}$	$T_{9,6} = \{f_n\}$	$T_{9,7} = \{f_{15}\}$	$T_{9,8} = \{f_n\}$	$T_{9,9} = \{f_n\}$	$T_{9,10} = \{f_{14}\}$
$T_{10,0} = \{f_n\}$	$T_{10,1} = \{f_n\}$	$T_{10,2} = \{f_n\}$	$T_{10,3} = \{f_n\}$	$T_{10,4} = \{f_n\}$	$T_{10,5} = \{f_n\}$	$T_{10,6} = \{f_n\}$	$T_{10,7} = \{f_n\}$	$T_{10,8} = \{f_n\}$	$T_{10,9} = \{f_{16}\}$	$T_{10,10} = \{f_n\}$

References

1. Wang, J.; Tang, Y.; Kavalen, J.; Abdelzaher, A.F.; Pandit, S.P. Autonomous UAV Swarm: Behavior Generation and Simulation. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Pearl Street Dallas, TX, USA, 12–15 June 2018; pp. 1–8. [CrossRef]
2. Kolling, A.; Walker, P.; Chakraborty, N.; Sycara, K.; Lewis, M. Human Interaction with Robot Swarms: A Survey. *IEEE Trans. Hum. Mach. Syst.* **2015**, *46*, 1–18. [CrossRef]
3. Hamann, H. *Swarm Robotics: A Formal Approach*; Springer International Publishing AG: Cham, Switzerland, 2018. [CrossRef]
4. Sahin, E. Swarm Robotics: From Sources of Inspiration to Domains of Application. *Swarm Robot.* **2005**, *3342*, 10–20. [CrossRef]
5. Nedjah, N.; Junior, L.S. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm Evol. Comput.* **2019**, *50*, 100565. [CrossRef]
6. Couzin, I.D.; Krause, J.; James, R.; Ruxton, G.D.; Franks, N.R. Collective Memory and Spatial Sorting in Animal Groups. *J. Theor. Biol.* **2002**, *218*, 1–11. [CrossRef] [PubMed]
7. Floreano, D.; Mattiussi, C. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*; The MIT Press: Cambridge, MA, USA, 2008.
8. Byrski, A.; Kisiel-Dorohinicki, M. *Evolutionary Multi-Agent Systems From Inspirations to Applications*; Springer: Cham, Switzerland, 2017. [CrossRef]
9. Bullo, F.; Cortés, J.; Martínez, S. *Distributed Control of Robotic Networks*; Applied Mathematics Series. Available online: http://coordinationbook.info (accessed on 15 January 2021).
10. Schweitzer, F. Brownian Agent Models for Swarm and Chemotactic Interaction Brownian Agents. In Proceedings of the Fifth German Workshop on Artificial Life, Lübeck, Germany, 18–20 March 2002; pp. 181–190.
11. Abelson, H.; Allen, D.; Coore, D.; Hanson, C.; Homsy, G.; Knight, T.F.T.; Nagpal, R.; Rauch, E.; Sussman, G.; Weiss, R. Amorphous Computing. *Commun. ACM* **2001**, *43*. [CrossRef]
12. Bachrach, J.; McLurkin, J.; Grue, A. Protoswarm: A Language for Programming Multi-Robot Systems Using the Amorphous Medium Abstraction. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '08, Estoril, Portugal, 12–16 May 2008; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2008; Volume 3, pp. 1175–1178.

13. Beal, J.; Viroli, M. Aggregate Programming: From Foundations to Applications. In *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems, Proceedings of the 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2016, Bertinoro, Italy, 20–24 June 2016*; Springer International Publishing: Cham, Switzerland, 2016; pp. 233–260. [CrossRef]
14. Damiani, F.; Viroli, M.; Pianini, D.; Beal, J. Code Mobility Meets Self-organisation: A Higher-Order Calculus of Computational Fields. In *Formal Techniques for Distributed Objects, Components, and Systems*; Graf, S., Viswanathan, M., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 113–128.
15. Pianini, D.; Viroli, M.; Beal, J. Protelis: Practical aggregate programming. In Proceedings of the ACM Symposium on Applied Computing, Salamanca, Spain, 13–17 April 2015; pp. 1846–1853.
16. Çelikkanat, H.; Sahin, E. Steering self-organized robot flocks through externally guided individuals. *Neural Comput. Appl.* **2010**, *19*, 849–865. [CrossRef]
17. Dudek, G.; Jenkin, M.; Milios, E.; Wilkes, D. A taxonomy for swarm robots. In Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93), Yokohama, Japan, 26–30 July 1993; Volume 1, pp. 441–447. [CrossRef]
18. Bayindir, L.; Sahin, E. A review of studies in swarm robotics. *Turk. J. Electr. Eng. Comput. Sci.* **2007**, *15*, 115–147.
19. Brambilla, M.; Ferrante, E.; Birattari, M.; Dorigo, M. Swarm Robotics: A Review from the Swarm Engineering Perspective. *Swarm Intell.* **2013**, *7*, 1–41. [CrossRef]
20. Mermoud, G.; Upadhyay, U.; Evans, W.C.; Martinoli, A. Top-Down vs. Bottom-Up Model-Based Methodologies for Distributed Control: A Comparative Experimental Study. In *Experimental Robotics: The 12th International Symposium on Experimental Robotics*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 615–629. [CrossRef]
21. Dorigo, M.; Tuci, E.; Groß, R.; Trianni, V.; Labella, T.H.; Nouyan, S.; Ampatzis, C.; Deneubourg, J.L.; Baldassarre, G.; Nolfi, S.; et al. The SWARM-BOTS Project. In *Swarm Robotics*; Şahin, E., Spears, W.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 31–44.
22. Dorigo, M. Swarm-Bots and Swarmanoid: Two Experiments in Embodied Swarm Intelligence. In Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Milan, Italy, 15–18 September 2009; Volume 2, pp. 2–3. [CrossRef]
23. Rubenstein, M.; Ahler, C.; Hoff, N.; Cabrera, A.; Nagpal, R. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robot. Auton. Syst.* **2014**, *62*, 966–975. [CrossRef]
24. Pinciroli, C.; Trianni, V.; O'Grady, R.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Di Caro, G.; Ducatelle, F.; et al. ARGoS: A Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intell.* **2012**, *6*, 271–295. [CrossRef]
25. Rohmer, E.; Singh, S.P.N.; Freese, M. CoppeliaSim (formerly V-REP): A Versatile and Scalable Robot Simulation Framework. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013.
26. Milner, R. *The Space and Motion of Communicating Agents*; Cambridge University Press: Cambridge, UK, 2009; Volume 20. [CrossRef]
27. Krivine, J.; Milner, R.; Troina, A. Stochastic Bigraphs. *Electron. Notes Theor. Comput. Sci.* **2008**, *218*, 73–96. [CrossRef]
28. Sevegnani, M.; Calder, M. Bigraphs with sharing. *Theor. Comput. Sci.* **2015**, *577*, 43–73. [CrossRef]
29. Benford, S.; Calder, M.; Rodden, T.; Sevegnani, M. On Lions, Impala, and Bigraphs: Modelling Interactions in Physical/Virtual Spaces. *ACM Trans. Comput. Hum. Interact.* **2016**, *23*. [CrossRef]
30. Mansutti, A.; Miculan, M.; Peressotti, M. Multi-agent Systems Design and Prototyping with Bigraphical Reactive Systems. In *Distributed Applications and Interoperable Systems*; Magoutis, K., Pietzuch, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 201–208.
31. Taki, A.; Dib, E.; Sahnoun, Z. Formal Specification of Multi-Agent System Architecture. In Proceedings of the ICAASE 2014 International Conference on Advanced Aspects of Software Engineering, Constantine, Algeria, 2–4 November 2014.
32. Pereira, E.; Potiron, C.; Kirsch, C.M.; Sengupta, R. Modeling and controlling the structure of heterogeneous mobile robotic systems: A bigactor approach. In Proceedings of the 2013 IEEE International Systems Conference (SysCon), Orlando, FL, USA, 15–18 April 2013; pp. 442–447. [CrossRef]
33. Agha, G. *Actors: A Model of Concurrent Computation in Distributed Systems*; MIT Press: Cambridge, MA, USA, 1986.
34. Gassara, A.; Bouassida Rodriguez, I.; Jmaiel, M.; Drira, K. Executing bigraphical reactive systems. *Discret. Appl. Math.* **2019**, *253*, 73–92. [CrossRef]
35. Sevegnani, M.; Calder, M. BigraphER: Rewriting and Analysis Engine for Bigraphs. In *Computer Aided Verification*; Chaudhuri, S., Farzan, A., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 494–501.
36. Perrone, G.; Debois, S.; Hildebrandt, T. A model checker for Bigraphs. In Proceedings of the ACM Symposium on Applied Computing, Riva, Trento, Italy, 26–30 March 2012. [CrossRef]
37. Brambilla, M.; Brutschy, A.; Dorigo, M.; Birattari, M. Property-Driven Design for Robot Swarms: A Design Method Based on Prescriptive Modeling and Model Checking. *ACM Trans. Auton. Adapt. Syst.* **2014**, *9*. [CrossRef]
38. Cybulski, P. Tracking_Bigraph Library. Available online: https://github.com/zajer/trs (accessed on 15 January 2021).
39. Cybulski, P. Exemplary Implementation of A Software to Transform TBRS into State Space. Available online: https://github.com/zajer/trs-ssp-bridge (accessed on 5 January 2021).
40. Cybulski, P. A Library for Calculating State Space Policies. Available online: https://github.com/zajer/state_space_policy (accessed on 15 January 2021).