Process Algebras and Flocks of Birds

Rocco De Nicola[1], Luca Di Stefano[2], Omar Inverso[3], and Serenella Valiani[1](✉)

[1] IMT School of Advanced Studies, Lucca, Italy
[2] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, Grenoble, France
[3] Gran Sasso Science Institute (GSSI), L'Aquila, Italy

Abstract. Many natural and artificial systems studied across a variety of disciplines, from biology to social sciences, consist of relatively simple agents with a partial knowledge of the system as a whole, where complex collective dynamics that are difficult to anticipate emerge from local interaction. We argue how formal methods broadly understood can be of assistance in such studies with a systematic approach to specification and analysis. To convey our argument, we elaborate a proof of concept inspired from an instance of emergent behaviour commonly observed in flocks of birds.

1 Introduction

Sophisticated *collective* dynamics can be observed in a variety of biological systems [18], such as herds of animals [25], colonies of insects [28], flocks of birds and schools of fish [23], but also in artificial systems such as political parties [27], smart cities [29], cyber-physical systems, and many others [8, 15, 21]. The study of such systems poses several challenges, such as intuitive specification and fast validation of different hypotheses on so-called *emergent behaviour* and other complex properties.

In this paper, we argue that concepts, methods, and tools from the the wider area of formal methods, to which Frits Vaandrager has dedicated most of his research efforts, can be of assistance in such activities. In particular, with the right ingredients, an integrated approach to formal specification and verification can open up to seemingly distant disciplines, where there could be plenty to be gained. The right ingredients here consist of a domain-specific formal language and effective verification procedures.

We consider a well-known example of collective behaviour known as *flocking*, that spontaneously emerges from the movement of birds in a flock. It is a fascinating natural phenomenon studied in a variety of disciplines, including ethology [22], optimization [1], economics [11], biology [18], and many others.

Flocking was considered as the combined effect of conflicting forces in the 1950s by Emlen, who proposed a model where an attractive force, which causes the birds to move closer to each other, is combined with a repulsive force that limits the size of the flock [17]. In the late 1980s, Reynolds refined this concept by introducing three separate rules, namely *cohesion*, *alignment*, and *separation*,

where flockmates move closer to each other when far apart, adapt their movements according to those of their neighbours, and avoid collisions by keeping a minimum distance from each other, respectively [23]. In practice, the combined effect of Reynold's rules can generate collective patterns of movement that resemble those of flocks of birds in the nature. Reynold's flocking model is an interesting example of bottom-up modelling of sophisticated collective behaviour via simple local rules. Indeed, the idea that collective behaviour can be expressed in terms of local interactions in a natural way is also backed up by more recent studies, from biology [18, 12] to physics [3, 4].

Elaborating an accurate model of flocking is outside the scope of this paper. Rather, we are interested in the study of minimalistic models that can mimic the dynamics described above at least in part. We focus on cohesion, which is commonly acknowledged as a core property of flocking [26, 13, 28]. Cohesion is usually defined based on the capability of each bird to determine a flock centroid (e.g., in Reynold's model, the barycentre of birds in the cohesion zone [23]). We would like to see whether it would be possible to achieve something similar to cohesion with minimal specifications, for instance by simply having birds approach pairwise non-deterministically.

We thus develop an initial model of flocking behaviour using a process algebra for collective systems [14] (Sect. 2). We carry out the analysis of cohesion using sequential emulation [16], spotting a corner case in the specifications which prevents cohesion (Sect. 3). We use the counterexample produced by our analysis to refine the specifications, and re-analyse cohesion (Sect. 4). We report some final considerations in Sect. 5.

2 A simplified model of a flocking behaviour

Let us now consider a minimalistic model of flocking behaviour, where each bird b looks at another bird a in the flock, estimates the future position of a based on a's current movement, and aims at moving towards that position (Fig. 2). To account for inertia, the new direction of b is averaged with its previous direction. To avoid collisions, in case the position where b wants to move is already occupied by another flockmate, b slows down. Assuming that all birds in the flock behave like this, we would like to know whether such rules would be sufficient to achieve cohesion.

Cohesion is usually defined based on the capability of each bird to determine a flock centroid (e.g., in Reynold's model, the barycentre of birds in the cohesion zone [23]). In our model, instead, a bird tries to approach a point where another bird in the system is likely going to be in the future.

We formalise the description given above by relying on the process description language LAbS [14]. A simplified version of the formal specifications is shown in Algorithm 1. Lines 2–6 describe the *interface*, i.e. the set of features, or *attributes*, that a bird exposes to the rest of the system. Here, the interface contains two attributes `x` and `y` describing the position of a bird on a two-dimensional grid, and two attributes `dir_x` and `dir_y` representing the movement vectors of the bird

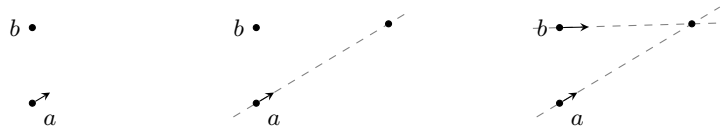

Fig. 2: Bird b targets bird a (a), looks at a's direction (b), and gets closer (c).

along the two axes. Each attribute has a range of feasible values. The position attributes may range over the interval $[0, G[$ and the direction attributes over $[-D, D+1[$. G and D are two parameters that respectively denote the size of the grid where the birds move, and the maximum length of the movement vectors (see Fig. 5 for the possible directions with $D = 1$). The symbol $\leftarrow$ denotes assignments to attributes. To model non-deterministic initialisation, attributes are assigned ranges of values rather than specific ones. Each agent also has an implicit attribute `id`, which is a unique identifier between 0 and the number of agents in the system.

The behaviour of each bird is defined in lines 8–27. The recursive definition at line 8 indicates that each bird repeatedly performs the same actions, given in process **Move**. This process is in turn defined as a sequence of assignments. Please note the symbol $\coloneqq$ that denotes assignments to local variables, and the enclosing curly braces that enforce atomicity.

The **Move** process consists of two parts. The first part (lines 11–19) implements the mechanism presented at the beginning of the section (Fig. 2). First, we non-deterministically select one agent by means of the *pick 1* command and assign it to a variable p (line 11). Then, we check whether this agent is *isolated* or not. We define p to be isolated when its distance from every other agent is larger than a parameter δ (line 12). Note that, in general, an attribute name decorated with an id (e.g., $\mathtt{x}_p$) evaluates to the value of the attribute for the agent with the given id. If the selected bird p is not isolated, the bird will approach it; otherwise, the bird will keep moving in its current direction. Also note that we define the distance operator $d(\cdot)$ at line 12 as the *Manhattan distance*, or ℓ_1-norm [7]: the distance between two points is the sum of the absolute differences between their components. Specifically, given two points $\mathbf{b}, \mathbf{p}$ in a two-dimensional space, we have $d((x_\mathbf{b}, y_\mathbf{b}), (x_\mathbf{p}, y_\mathbf{p})) = |x_\mathbf{b} - x_\mathbf{p}| + |y_\mathbf{b} - y_\mathbf{p}|$, which corresponds to the combined length of the segments shown in Fig. 3. Starting from line 14 the bird estimates the future position (ax, ay) of the agent *appId* to approach by multiplying its direction vector by a parameter ω. It then approximates a movement vector *adir* towards that position by comparing $(\mathtt{x}, \mathtt{y})$ and (ax, ay) componentwise with a tolerance parameter ε. We only report the instructions for the x component of the vector; the y component is computed similarly.

In the second part of the **Move** process (lines 22–26), the bird updates its own attributes. Specifically, the bird's new direction is the average between the previous one and the vector *adir* computed beforehand. Please note that the

Listing 1: Initial specifications for a flock of birds.

```
1  agent Bird {
2      Interface =
3          x ← 0..G;
4          y ← 0..G;
5          dir_x ← −D..D + 1;
6          dir_y ← −D..D + 1
7
8      Behaviour = Move; Behaviour
9
10     Move = {
11         p := pick 1;
12         pIsIsolated := forall Bird b, b ≠ p ⇒ d((x_p, y_p), (x_b, y_b)) > δ;
13         appId := if pIsIsolated then id else p;
14         ax := x_appId + ω · dir_x_appId;
15         sgn_x := if x > ax then 1 else −1;
16         adir_x :=  if a = id then
17                        | dir_x
18                    else
19                        ⌊ if |x − ax| < ε then 0 else sgn_x · D;
20         # assign ay, sgn_y, adir_y as above
21
22         dir_x ← (dir_x + adir_x)/2;
23         dir_y ← (dir_y + adir_y)/2;
24         posIsFree := forall Bird b, (x_b ≠ x + dir_x) ∨ (y_b ≠ y + dir_y);
25         x ← if posIsFree then x + dir_x else x
26         y ← if posIsFree then y + dir_y else y
27     }
28 }
```

division used here is an integer division with rounding. Finally, the bird checks whether the cell it would reach by moving along its new direction is free: if so, the bird moves there by updating its attributes `x` and `y`; otherwise, it stays in its current cell (lines 24–26).

3 Analysis of cohesion

We now carry out the analysis of cohesion for the model of flocking behaviour given in the previous section. The key element of our verification flow is a symbolic encoding of the specifications into a sequential imperative program, which we call an *emulation program* [16]. The encoding reduces the problem of checking whether the system satisfies the given property to checking reachability in the emulation program. This has the twofold advantage of detaching the specification language from the verification technique, and allowing to automatically re-use program analysis tools for general-purpose languages.

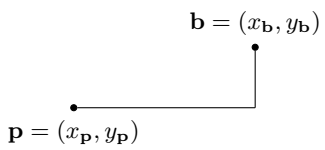

Fig. 3: Manhattan distance between two points in two dimensions.

The emulation program uses a minimal set of features (i.e., loops and statically-sized arrays), and can be concretised with limited effort into different target languages, depending on the verification technology of preference; it also embeds an explicit scheduler, which allows to apply specific scheduling policies. We target the C language and rely on bounded model checking [9] for the actual analysis; we choose round-robin scheduling, i.e., agents perform their actions in a round-robin fashion. We call *epoch* an execution fragment in which every agent in the system performs precisely one action. This allows us to consider verification bounds in terms of epochs. The verification flow described above is implemented in our prototype tool SLiVER[4], that takes care of generating the emulation program from the specifications of the system under analysis, instrumenting the emulation program for verification to be carried out by the back end model checker, and translating any counterexample from the model checker into a human-readable output with respect to the initial system specifications.

In order to assess cohesion, we set up an scenario with two separate groups of birds positioned at a certain distance from each other (Fig. 4), and check whether, given enough epochs for the system to evolve, the two groups end up forming a single flock. We thus instantiate the system of Listing 1 with four agents, a grid of size $G = 1024$ (lines 3–4), movement vectors of max modulo $D = 1$ for the possible directions of agents (lines 5–6), a sensitivity $\omega = 10$ to estimate the future position of the bird to approach (line 14), a distance $\delta = 32$ to determine whether an agent is isolated (line 12), and a tolerance parameter $\varepsilon = 5$ to approximate the approach vector (line 19). We non-deterministically position the two groups of birds into two smaller sub-grids of size 9×9, the birds in the left-hand group oriented bottom to top, and those in the right-hand group oriented right to left. The two regions are 40 cells apart, therefore the Manhattan distance between any two birds from different groups is initially at most 76 cells. Figure 6a shows a feasible initial state under these constraints. We enforce these constraints by specifying them as quantified predicates in a dedicated section of the specifications (Listing 2).

With the above set up, we use our prototype to check whether, after B steps, every execution of the system reaches a state where all birds are at most k cells apart (lower values of k indicating a more compact flock and thus a stronger cohesion). To express this property, we decorate the specifications of Listing 1 as shown in Listing 3. Since birds are initially not farther than 76 cells, we start checking the property for a cohesion distance k of 75, to check whether

[4] The tool is available at https://github.com/labs-lang/sliver.

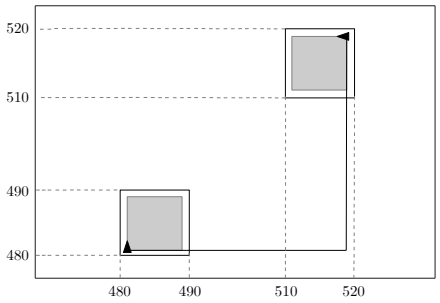

Fig. 4: Initial areas, in grey, where agents can position themselves.

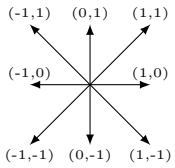

Fig. 5: Possible directions of a bird with movement vectors of max. size $D=1$.

the birds can get barely closer together than in the initial state. Indeed, our tool immediately produces a counterexample showing that the specifications of Listing 1 fail to guarantee even such minimal degree of cohesion.

The counterexample is shown in Fig. 7. Intuitively, since each bird keeps approaching a bird within the same group, the groups will stay separate indefinitely if they don't meet by accident, and thus the flock will never achieve cohesion. Figure 6a shows the initial state of the system. During the first epoch, first the 1-orange and 2-blue agents choose to approach the 3-red and 4-green ones, respectively, altering their direction accordingly (Figures 6b). Then, the 3-red and 4-green agents make the symmetrical choice (Figure 6c). At this point, the two subgroups have parallel direction vectors and the cohesion property is still unsatisfied, as the red agent is more than 75 cells apart from the green one. From now on, agents in each subgroup keep selecting each other as the agent to approach, meaning that the subgroups keep moving parallel to each other and never achieve the desired degree of cohesion (Figure 6d).

4 Revising the model

The counterexample obtained in Sect. 3 shows that when adopting the behaviour of Listing 1 the birds may never achieve cohesion, as they will completely ignore the other birds outside their group. In this section we modify the specification of Listing 1 to address this problem so that each bird can also approach agents outside its own group. We then repeat the analysis to check whether the revised specifications improve cohesion.

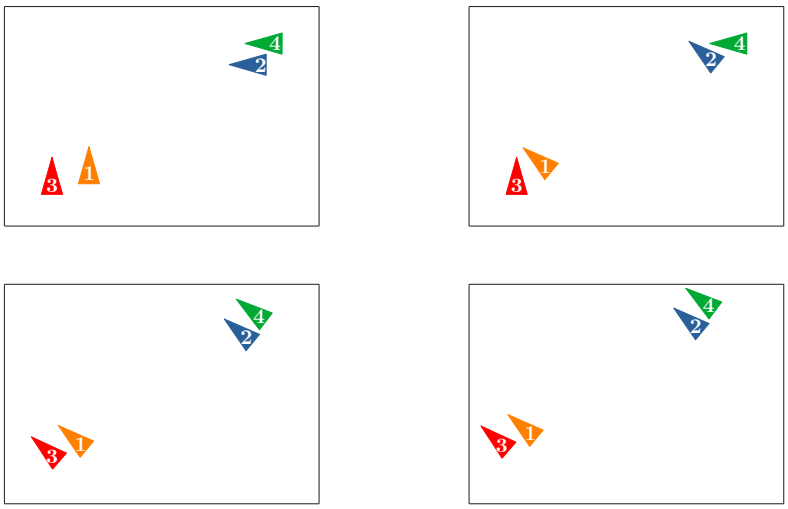

Fig. 7: Two groups of birds failing to achieve cohesion.

Intuitively, as a possible way to improve cohesion, a bird should be able to alternately approach other birds from his own group and from the other group. To accommodate this, we change the specifications as shown in Listing 4. An attribute `groupId` initialized to either 0 or 1 keeps track which group a bird initially belongs to. Another attribute, `check`, is initialized to 0 and is used to guide the selection of the bird to be approached (lines 4–5). Finally, line 11 of Listing 1 is replaced by lines 9–10 of Listing 4. After this change, the non-deterministic selection of the bird p to be approached is constrained by a predicate, introduced by the keyword `where`. This predicate states that, if the attribute `check` is currently set to 0, the agent may pick any bird indiscriminately. However, if `check` is set to 1, then the agent must pick a bird whose `groupId` is different from its own (line 9). Then, the agent flips the value of `check` (line 10). This means that a bird will necessarily pick somebody outside its own initial group at least every other epoch.

We repeat the same experiment described in Sect. 3 on the specifications revised as above to verify flocks of 4, 6, and 8 agents with the same parameters listed in Section 3, increasing the verification bound until obtaining a positive verdict. Figure 8 reports the minimum number of epochs needed to reach a positive verdict for varying systems and values of k. The number of epochs grows linearly as the cohesion distance k decreases and does not blow up as the number of birds increases, suggesting that achieving cohesion does not become particularly harder for larger flocks, at least not for such simple specifications.

Figure 9 report additional measurements on the amount of time and memory needed to obtain the positive verdicts reported in Fig. 8, where we can observe

Listing 2: Initial scenario with two separate groups of birds.

```
1 assume {
2     DifferentPositions =
        forall Bird a, forall Bird b, a = b ∨ x_a ≠ x_b ∨ y_a ≠ y_b
3     GridLeft =
        forall Bird b, (id_b mod 2 = 0) ∨ ((480 < x_b < 490) ∧ (480 < y_b < 490))
4     GridRight =
        forall Bird b, (id_b mod 2 ≠ 0) ∨ ((510 < x_b < 520) ∧ (510 < y_b < 520))
5     AlignmentLeft = forall Bird b, (id_b mod 2 = 0) ∨ (dirx_b = 0 ∧ diry = 1)
6     AlignmentRight =
        forall Bird b, (id_b mod 2 ≠ 0) ∨ (dirx_b = −1 ∧ diry = 0)
7 }
```

Listing 3: Cohesion property.

```
1 check {
2     Cohesion = finally forall Bird b, forall Bird c, d((x_b, y_b), (x_c, y_c)) < k
3 }
```

that the performance quickly degenerates when increasing the number of birds and of epochs; the model checker must in fact exhaustively explore the state space up to a bound which is given by the number of epochs multiplied by the number of agents. Changing the back end technology can affect the efficiency of analysis significantly [16], but comparing different techniques is outside the scope of this paper.

We performed all the experiments in a virtualized environment on a dedicated machine running 64-bit GNU/Linux with kernel 5.4.0 and equipped with four 2-GHz Xeon E7-4830v4 10-core processors and 512 GB of physical memory.

5 Conclusion

The main point we intend to make with this paper is that existing methodologies, techniques, and tools from the wider area of formal methods, when appropriately combined, can be of assistance in the study of different classes of so-called collective systems of interest in a variety of disciplines. To support our argument, we have shown how formal languages and modern verification procedures can be combined to study the behaviour of flocks of birds. The technical contribution of this paper is clearly only a proof of concept to support our argument, but in our opinion points out relevant research directions which it may be worthwhile pursuing.

With respect to the specific scenario considered in this paper, we plan to devise further refined models of flocking behaviour and formally verify different properties, possibly working on the back end verification technique to improve efficiency, considering distributed analysis or large-scale simulation with com-

Listing 4: Revised version of the specifications in Listing 1.

```
1  agent Bird {
2      Interface =
3          ...
4          groupId ← 0..2;
5          check ← 0
6
7      Behaviour = Move; Behaviour
8      Move = {
9          p := pick 1 where (check = 0) ∨ (groupId ≠ groupId_p);
10         check ← (check + 1) mod 2;
11         ...
12     }
13 }
```

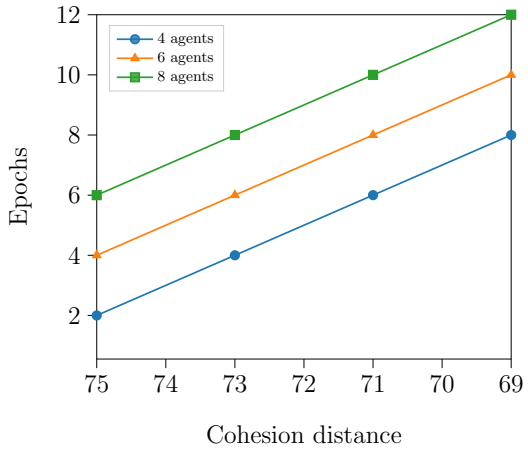

Fig. 8: Number of epochs to achieve cohesion at different distances.

puting clusters. We certainly plan to apply our methodology to other classes of systems, either artificial or natural, and would like to try to interact with researchers from different areas in the long run.

From a technical standpoint, our approach can be improved in several ways. With respect to scalability, techniques for *parameterized* model checking may help, as they would enable us to demonstrate that a property holds for all systems larger than a threshold. So far, these techniques have been demonstrated on models and languages with limited agent capabilities [19, 5, 20]; the question whether they may be adapted to more high-level languages such as LAbS is open. Orthogonal approaches such as symmetry reduction and partial order reduction [10, 2] could facilitate the verification of larger systems, but their integration in our verification flow might require considerable efforts. Lastly, in this paper we have only experimented with bounded analysis. Indeed, our verifica-

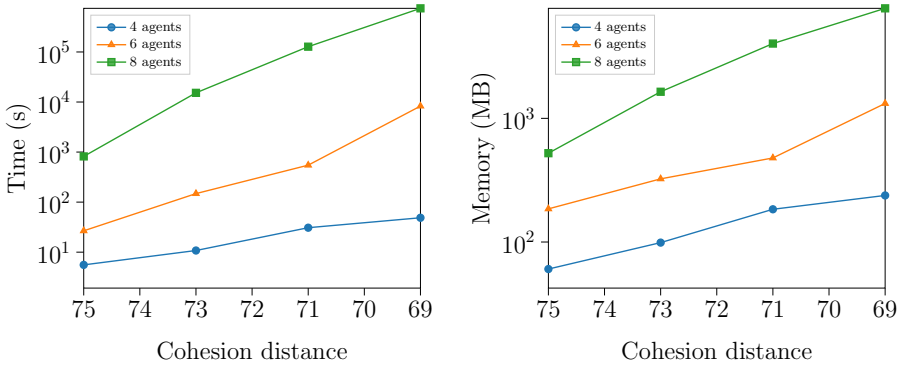

Fig. 9: Resources needed to verify cohesion varying cohesion distance.

tion workflow can accommodate other kinds of back end technologies. It would be interesting to experiment with unbounded verification of properties using state-of-the-art inductive techniques such as k-induction [24] or PDR [6].

References

1. Alaliyat, S., Yndestad, H., Sanfilippo, F.: Optimisation of boids swarm model based on genetic algorithm and particle swarm optimisation algorithm (comparative study). In: Squazzoni, F., Baronio, F., Archetti, C., Castellani, M. (eds.) 28th European Conference on Modelling and Simulation, ECMS 2014, Brescia, Italy, May 27-30, 2014. pp. 643–650. European Council for Modeling and Simulation (2014). https://doi.org/10.7148/2014-0643, https://doi.org/10.7148/2014-0643
2. Alur, R., Brayton, R.K., Henzinger, T.A., Qadeer, S., Rajamani, S.K.: Partial-order reduction in symbolic state space exploration. In: Grumberg, O. (ed.) 9th International Conference on Computer Aided Verification (CAV). LNCS, vol. 1254, pp. 340–351. Springer, Haifa, Israel (Jun 1997). https://doi.org/10.1007/3-540-63166-6_34
3. Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Lecomte, V., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., Zdravkovic, V.: Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. Proceedings of the National Academy of Sciences **105**(4), 1232–1237 (2008). https://doi.org/10.1073/pnas.0711437105, https://www.pnas.org/doi/abs/10.1073/pnas.0711437105
4. Bialek, W., Cavagna, A., Giardina, I., Mora, T., Silvestri, E., Viale, M., Walczak, A.M.: Statistical mechanics for natural flocks of birds. Proceedings of the National Academy of Sciences **109**(13), 4786–4791 (2012)
5. Blondin, M., Esparza, J., Jaax, S.: Peregrine: A tool for the analysis of population protocols. In: Chockler, H., Weissenbacher, G. (eds.) 30th International Conference on Computer Aided Verification (CAV). LNCS, vol. 10981, pp. 604–611. Springer (2018). https://doi.org/10.1007/978-3-319-96145-3_34
6. Bradley, A.R.: SAT-Based Model Checking without Unrolling. In: Jhala, R., Schmidt, D.A. (eds.) 12th International Conference on Verification, Model Check-

ing, and Abstract Interpretation (VMCAI). LNCS, vol. 6538, pp. 70–87. Springer (2011). https://doi.org/10.1007/978-3-642-18275-4_7
7. Brezis, H., Brézis, H.: Functional analysis, Sobolev spaces and partial differential equations, vol. 2. Springer (2011)
8. Casadei, R., Viroli, M.: Programming actor-based collective adaptive systems. In: Programming with Actors, pp. 94–122. Springer (2018)
9. Clarke, E., Kroening, D., Lerda, F.: A tool for checking ANSI-C programs. In: Jensen, K., Podelski, A. (eds.) 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). pp. 168–176. LNCS, Springer (2004). https://doi.org/10.1007/978-3-540-24730-2_15
10. Clarke, E.M., Emerson, E.A., Jha, S., Sistla, A.P.: Symmetry reductions in model checking. In: Hu, A.J., Vardi, M.Y. (eds.) 10th International Conference on Computer Aided Verification (CAV). LNCS, vol. 1427, pp. 147–158. Springer, Vancouver, BC, Canada (1998). https://doi.org/10.1007/BFb0028741
11. Cont, R., Bouchaud, J.P.: Herd behavior and aggregate fluctuations in financial markets. Macroeconomic dynamics **4**(2), 170–196 (2000)
12. Couzin, I.D., Krause, J., James, R., Ruxton, G.D., Franks, N.R.: Collective memory and spatial sorting in animal groups. Journal of theoretical biology **218**(1), 1–11 (2002)
13. Craig, W.: Appetites and aversions as constituents of instincts. The Biological Bulletin **34**(2), 91–107 (1918)
14. De Nicola, R., Di Stefano, L., Inverso, O.: Multi-agent systems with virtual stigmergy. Science of Computer Programming **187**, 102345 (2020). https://doi.org/10.1016/j.scico.2019.102345
15. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétien, L.: The dynamics of collective sorting robot-like ants and ant-like robots. In: From animals to animats: proceedings of the first international conference on simulation of adaptive behavior. pp. 356–365 (1991)
16. Di Stefano, L., De Nicola, R., Inverso, O.: Verification of distributed systems via sequential emulation. ACM Transaction on Software Engineering and Methodology **31**(3) (2022). https://doi.org/10.1145/3490387
17. Emlen, J.T.: Flocking behavior in birds. The Auk **69**(2), 160–170 (1952)
18. Grégoire, G., Chaté, H., Tu, Y.: Moving and staying together without a leader. Physica D: Nonlinear Phenomena **181**(3-4), 157–170 (2003)
19. Konnov, I., Veith, H., Widder, J.: SMT and POR beat counter abstraction: Parameterized model checking of threshold-based distributed algorithms. In: Kroening, D., Pasareanu, C.S. (eds.) 27th International Conference on Computer Aided Verification (CAV). LNCS, vol. 9206, pp. 85–102. Springer, San Francisco, CA, USA (Jul 2015). https://doi.org/10.1007/978-3-319-21690-4_6
20. Kouvaros, P., Lomuscio, A.: A counter abstraction technique for the verification of robot swarms. In: Bonet, B., Koenig, S. (eds.) 29th Conference on Artificial Intelligence (AAAI). pp. 2081–2088. AAAI (2015)
21. Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. Robotics and autonomous systems **30**(1-2), 85–101 (2000)
22. Norris, K.S., Schilt, C.R.: Cooperative societies in three-dimensional space: on the origins of aggregations, flocks, and schools, with special reference to dolphins and fish. Ethology and Sociobiology **9**(2-4), 149–179 (1988)
23. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Stone, M.C. (ed.) Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA,

July 27-31, 1987. pp. 25–34. ACM (1987). https://doi.org/10.1145/37401.37406, https://doi.org/10.1145/37401.37406
24. Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-Solver. In: Jr., W.A.H., Johnson, S.D. (eds.) 3rd International Conference on Formal Methods in Computer-Aided Design (FMCAD). LNCS, vol. 1954, pp. 108–125. Springer (2000). https://doi.org/10.1007/3-540-40922-X_8
25. Sumpter, D.J.: The principles of collective animal behaviour. Philosophical transactions of the royal society B: Biological Sciences **361**(1465), 5–22 (2006)
26. Trotter, W.: Instincts of the Herd in Peace and War. TF Unwin Limited (1920)
27. Valentini, G., Hamann, H., Dorigo, M., et al.: Self-organized collective decision making: the weighted voter model. In: AAMAS. pp. 45–52 (2014)
28. Wheeler, W.M.: The social insects: their origin and evolution. Routledge (2015)
29. Zedadra, O., Guerrieri, A., Jouandeau, N., Spezzano, G., Seridi, H., Fortino, G.: Swarm intelligence and IoT-based smart cities: A review. In: Cicirelli, F., Guerrieri, A., Mastroianni, C., Spezzano, G., Vinci, A. (eds.) The Internet of Things for Smart Urban Ecosystems, pp. 177–200. Springer (2019)