Lensing: It is All About Perspective

Ada Diaconescu*, David King†, Kirstie Bellman‡, Christopher Landauer‡, Phyllis Nelson§
*Telecom Paris, IP Paris, LTCI, Paris, FR
†Air Force Institute of Technology, Dayton, Ohio, USA
‡ Topcy House Consulting, LA, USA
§ California State Polytechnic University, Pomona, CA, USA

***Abstract*—Complex Adaptive Systems (CAS) are difficult to understand and predict. Certain CAS phenomena can only be observed at certain abstraction levels – e.g., swarm dynamics cannot be analysed by only tracking one individual. In most current systems the "right" abstraction levels for each question asked, or each problem solved, are determined at design time. Yet, in adaptive self-integrating systems the abstraction level must be determined dynamically, depending on the questions encountered and the system context. This position paper aims to set-up the basis for a research initiative in this direction. It proposes the concept of '*lensing*' to tune a system's observation granularity in terms of spatial and temporal scope, and information detail. It further introduces *lens efficacy* and *efficiency* to evaluate a lens' ability to answer specific questions – a critical process for self-aware systems performing lensing at runtime. Finally, the paper illustrates the criticality of lensing in answering different kinds of questions via several examples of collective movement systems, including a Game of Life (GoL) Glider simulation.**

***Index Terms*—Keywords: lensing, abstraction, hierarchy, collective movement, complex adaptive systems, game of life**

I. Introduction

Answering questions about Complex Adaptive Systems (CAS) can be quite challenging. One of the hallmarks of complex systems is their hierarchical structure [1], [2], [3] (i.e., organized into multiple scales, abstraction levels or layers). The adaptive behavior of complex systems may lead to the occurrence, transformation or disappearance of certain abstraction levels within the system's hierarchy.

Different phenomena (i.e., properties of system state or behavior) can be observed more easily, or are only observable at certain system abstraction levels. Hence, answering different questions require system observation and analysis at suitable abstraction levels. In most current systems this "right level of abstraction" – with its terms, ontology, and semantics; as well as its associated analysis and reasoning processes – is all done at design time (e.g., for simulation purposes [4]). However, in self-adaptive and self-integrating systems this suitable abstraction level must be identified during runtime, based on the latest system structure; and then adapted dynamically to reflect further system changes. We can generalize this challenge by noting that the kind of processes required to answer questions about a system are also required to solve problems, accomplish tasks, or reach goals for that system (e.g., which set of components to employ and how to integrate them for obtaining a desired system state or behavior?).

In this position paper, we first illustrate how critical it is to choose the right abstraction level for different types of questions. We note that such questions will necessarily concern an Observer, which may be external or internal to the observed system (e.g., a human or another system attempting to understand or communicate with the observed system). We propose the concept of 'lensing' to refer to the Observer's runtime adaptation of the system's observed abstraction level.

We consider each 'lens' to represent the *granularity and range of observation* of a system, in terms of: i) *space* scope – i.e., how large is the observed system area; ii) *time* scope – i.e., how long is the observation interval or frequency; iii) *resolution* – the smallest distinguishable unit; and iv) *information* – i.e., which resolution, or amount of detail does an observation provide; which variables or aspects are in/out of the observation focus. Lensing emphasizes the Observer's quest to select the appropriate lens for answering specific questions, at runtime.

We take as a simple-to-understand example the questions that an Observer could associate with the movement of a herd of cattle. A first set of questions may include: Is there a herd? Are there several herds? Is the herd moving? How fast and in what direction? To answer these questions, the Observer does not need details of each animal – group-detection and the group centroid may suffice. Now consider a further question: How many cows are in the herd? To provide an estimate answer, the Observer needs to determine the herd area (space scope) and density (about how many animals per space unit). To provide an exact answer, the Observer needs sufficient detail to perceive each animal. This simple example shows how lensing helps answer different kinds of questions, with different accuracy: higher-level lenses for general herd movement questions; middle-level lenses for animal count estimates; and low-level lens for accurate animal counting.

We further aim to set the basis for evaluating the impacts of lensing on an Observer's ability to answer various questions. We highlight the following key lensing evaluation aspects:

- *Lens efficacy*: a lens is effective relative to a question about a system, if it allows to observe the system phenomena that are relevant for answering that question.
- *Lens efficiency*: while a range of lenses may be effective for a certain question, some lenses may require fewer resources for perceiving the phenomena relevant to that question, compared to the other lenses in the same effective range. We may then say that a lens is more efficient than another at answering a question.

To illustrate this basis for lensing evaluation, we introduce a more concrete simulation model for our herd example – a Glider in a Game of Life (GoL) Cellular Automata (CA) [5]. We then introduce various kinds of lenses and evaluate their efficacy and efficiency to answer the exemplified herd questions.

The paper's key contributions include:

- proposing the *lensing* concept for selecting suitable abstraction levels for answering questions at runtime;
- proposing lensing *efficacy* and *efficiency* as means to evaluate a lens' suitability to answer a question;
- illustrating the above concepts via several examples related to collective movement (i.e., herd and GoL Glider).

The above contributions set a preliminary basis for a more comprehensive theory of CAS lensing.

II. Background and Related Work

At its core, the proposed lensing notion focuses on an Observer's perception of an observed system. The Observer may be a human user, the system itself (i.e., self-aware system), or another artificial system (i.e., context-aware). Each lens may model the system at a different granularity (e.g., as a set of interconnected components, or as an integrated whole). To use the language of emergentists [1], lenses describe macro and micro-level behaviors and structures. In essence, lensing implies acquiring sets of abstractions and switching amongst them in an attempt to identify particular system properties.

A key viewpoint is that complexity is in the eye of the Observer and the Observer is the pivotal point of view [3]. As argued by Salthe [6], system representations of the world will never be complete. Hence, we assign abstractions to describe system structures and behaviors from different perspectives and granularities, to reach given objectives. These abstractions can take many forms – one of the most prevalent in the CAS literature advocates the composition and decomposition of hierarchical structures within self*-systems [2], [3], [6].

John Holland, an early and leading advocate for developing theories about self*-systems, included hierarchical organization as an integral part of CAS [1]. However, as Holland refined his theories on CAS, he advocated the use of the term *niche* to describe bounded areas of an ecosystem [7]. Niches represent a perspective lens that takes into account what agents or components work within a specific physical or metaphysical boundary. Niches model agent interactions and the types of signals that occur both internally and externally to the niche. A niche is not a hierarchical model. It is more of a flat, or horizontal, model that partitions the ecosystem into manageable, micro-level wholes connected only through signals that bind them together which can create temporal hierarchical, or even heterarchical, constructs.

Finding the "right" abstraction level for system modelling with respect to a given objective has been an ongoing problem, for both 'classic' and self-* systems. Although an Observer's models are system abstractions and can never provide complete system representations, they can still provide enough information to reach certain objectives, solve problems or answer questions (e.g., what is the system doing and what will it probably do in the future? from the Observer's perspective).

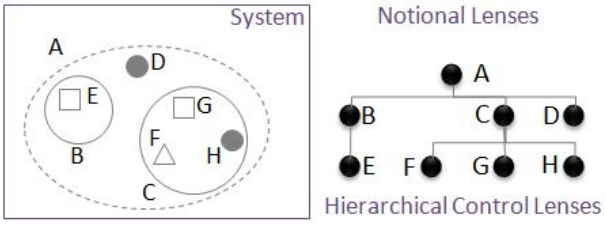


Fig. 1: Notional lensing of an agent-based system (extended from [6]) – Left: structural system lens based on composites; Right: hierarchical control lens based on causal relations; Additional lenses may include: Entity Lenses (Lens1: position of A; Lens2: pos. of B, C, D; Lens3: pos. of E, F, G, H); Comparative lenses (Lens4: C is larger than B and D); Containment lenses (Lens5: C contains 3 objects, B 1 object).

Lenses can provide system abstractions along diverse dimensions: e.g., spatial (scope and granularity), temporal (interval or frequency) or informational (aggregation, filtering, selection). The level of abstraction can then shift along each such dimension to produce further lenses. In the herd example (sec. I), the *spatial scope* represents the area of observation (e.g., including the entire herd or just part of it) and the *spatial resolution* the observation granularity of the scope (e.g., the entire herd or each cattle; or, the size of spatial units at which the spatial scope can be seen). The observation *interval* indicates the length of time over which the system is observed to detect something; and the *frequency* refers to the observation rate [8] (e.g., observing the herd every 5 minutes for 30 seconds to survey its movement trajectory). Finally, collected information can be transformed into higher-level concepts to provide meaningful semantics with fewer resources (e.g., only tracking the herd's centroid rather than following each cattle).

When studying a CAS, one develops not just one lens, but multiple lenses, each dedicated to presenting different representations of multiple system constructs at the macro and micro-levels. These lens representations can be aggregated into higher-level models or studied independently. A lens at a lower tier of a hierarchy could feed information to lenses *above*, thus acting as an information filter or aggregator, up and down and across the system structures. These lenses create different models, at different abstraction levels, via which an observer can adjust its viewpoint dynamically. Observers may also create lenses on-the-fly, by adding, subtracting, aggregating, disassembling lenses and lens outputs, at various granularity levels, to better understand the observed system.

Figure 1 provides an example of how one could construct various lenses to describe a composite system from various perspectives. To the left is a structural lens of an observed system. Structures B-C-D compose A, and as indicated by their boundaries, E composes B, F-G-H compose C, and D is a lone entity. We can then have different lenses for describing the system from various perspectives and abstraction levels.

To the right, we have a control lens, where A controls all the entities that compose it, and then, by extension, those smaller entities then control their component parts. Still, this hierarchical control lens only tells a part of the entire story. There are other potential lenses, e.g., comparative (B is smaller than C) or positional lenses (B, C & D reside within A).

Each lens presents different kinds of information about the system, in a unique manner. The abstractions used have a certain semantics for the Observer who can, over time, use lenses to describe the system structures and/or behaviors and, hopefully, to anticipate the changes it may undergo. It is important to note that this representation is a snapshot and we must not forget that we need a series of representations to derive meaning and function from observation.

The lensing concept is similar to that of *(multi-)scaling* in [9] [10], yet only focusing on system observation and analysis, rather than on entire feedback control loops. Lensing is also similar to notions of *coarse-graining* [11] or *blurring* [12], yet focusing on an Observer's quest to find and adapt appropriate abstraction levels for various questions.

Adaptive multi-level modeling is also similar to lensing. E.g., [13] propose to employ *adaptive abstraction* levels for offline system modelling. They aim to optimize simulation resources, by adjusting the abstraction level of the system's model during the simulation. Abstractions are adapted to maintain the desired properties (i.e., simulation objectives) while minimizing required resources, at any one time.

Comparatively, we focus on finding the best abstraction level (lens) for solving a particular problem, or answering a particular question about the system, at runtime (i.e., lens efficacy). The question of optimizing lensing efficiency is secondary. Our objective is compatible with that in [14], which proposes an approach for generating models at the appropriate temporal grain, or scale, for solving a given problem.

III. Game of Life: Glider Example

A. Simulation Overview

We start by implementing our theoretical example of group movement via a simple Cellular Automata (CA) simulation, using the Game of Life (GoL) rules [5] and an initial state that produces one Glider (i.e., Java simulation). The Glider features 4 states (Cf. top row of Fig. 2), which repeat via 4-step cycles as the simulation advances. All Glider states consist of 5 live cells that occupy a CA grid *patch* of 3x3 cells. At each cycle the patch that the Glider occupies moves down by one row and right by one column. Fig. 2 depicts the Glider movement over 3 cycles (12 simulation steps). It shows a part of the CA grid composed of 36 cells (6x6), which we divide into 4 patches (3x3). Over the 3 cycles, the Glider moves diagonally-down from Patch 1 (top-left corner) to Patch 4 (bottom-right corner). This shows the Glider's linear diagonal movement (at a 45 degree angle). In the presented experiments, we ran the simulation for 36 consecutive steps (9 cycles), using a CA grid of 40 x 80 cells in size.

We aim to show how viewing the Glider simulation via various lenses enables an external Observer to achieve various *tasks* or answer various *questions*. Via a bottom lens ($Lens0$) we can observe the simulation at the cell level (highest resolution available). We introduce two additional lenses that only observe the Glider simulation at the patch level (3x3 cells resolution). That is, the CA simulation grid is divided into 3x3 patches and each patch is mapped to a single cell in a higher lens. The cell in the higher lens is 'live' (colored) if the number of cells in the associated lower patch is higher or equal to an *abstraction-threshold*; and 'dead' otherwise (grey). We set the *abstraction-threshold* $= 1$ for $Lens1$; and $= 2$ for $Lens2$. Fig. 3 shows the corresponding states of $Lens1$ and $Lens2$ for the first 4 simulation steps.

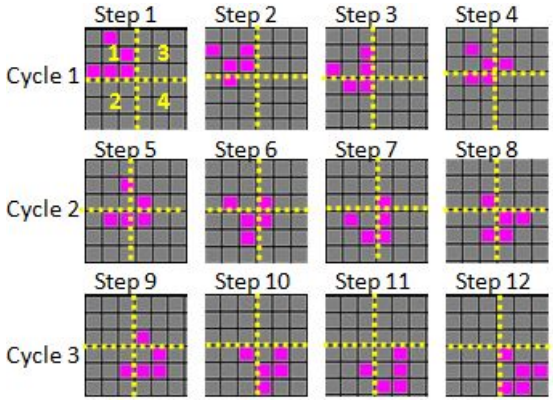


Fig. 2: Glider simulation ($Lens0$): 4 states in a Cycle (Step 1 to 4, top row). In each Cycle the Glider moves one row down (step1-step2) and one row right (step3-step4). Each Glider state fits within a patch of 3x3 cells (e.g., 4 patches shown above, in yellow – numbered 1, 2, 3, 4). Over 3 Cycles (12 Steps) the Glider moves diagonally from patch 1 to patch 4.

Fig. 3: Two Glider Lenses: map 3x3 cell patches from bottom lens to 1 higher cell in higher lens. Higher cell is live if the number of live cells in the lower patch is higher or equal to a threshold, which equals 1cell for $Lens1$ and 2cells for $Lens2$

B. Group detection and movement analysis

1) Questions: We ask the Observer the following questions (requiring corresponding *tasks*):

1) is there a Glider? task-1: detect a group of live cells;
2) is the Glider moving? task-2: detect group movement;
3) where is it going? task-3: estimate group trajectory.

2) Analysis method: We assume that the Observer has access to the lowest lens ($Lens0$), hence to the entire simulation state. To detect groups of cells (task-1), the Observer analyses the entire CA simulation and clusters into groups all live cells that neighbor each other directly – i.e., one Glider

detected at each simulation step. To detect movement (task-2), the Observer introduces a new lens that only keeps the centroid of each group ($Lens0 - core$). The Observer plots the group centroids on a 2D graph, roughly mapping onto the CA simulation grid. If a group's centroid is changing position in time then the Observer concludes that the group is moving. Finally, to determine a moving group's trajectory (task-3), the Observer uses linear regression to find the linear function that best approximates the group's plotted centroids. It is well-known that all sustained and unobstructed movement in GoL is linear with rational slopes.

In summary, to achieve its tasks, the Observer employs a *pair of lenses* (e.g., $Lens0$ and *Lens0-core*). It also applies the same process above (for task-1 to -3) when observing the Glider via the two higher lenses ($Lens1$ and $Lens2$); and their respective pairs (*Lens1-core* and *Lens2-core*).

3) Lensing Efficacy: Fig. 4 depicts the results when observing the Glider through *Lens0-core* and *Lens1-core*. *Lens2-core* produces results similar to *Lens1-core* (not shown for space limitations). In both cases, we note that the Observer can accomplish all tasks: detecting the Glider (task-1); detecting its movement (task-2); and determining its linear trajectory. The trajectory equations obtained are:

- via *Lens0-core*: $-0.99\,x + 38.47$,
- via *Lens1-core*: $-0.86\,x + 39.04$,
- via *Lens2-core*: $-0.8\,x + 39.31$,

which makes sense since all of these trajectories actually have a slope of -1.

4) Lensing Efficiency: We established above that all three pairs of lenses ($Lens0$, $Lens1$ and $Lens2$) were effective for reaching the Observer's movement analysis tasks. We can now estimate how efficient the three lenses pairs were in this respect. We first discuss the difference in resource usage and then compare the accuracy difference of the trajectories.

For group detection (task-1) the Observer must analyze the entire simulation grid (e.g., 40x80 = 3200 cells) when using $Lens0$. This can be reduced by up to 9 times when using the higher lens patches (3x3) – e.g., grid of 14x27 = 378 cells for $Lens1$ and $Lens2$, hence 8.5 times less than for $Lens0$.

For centroid plotting (part of task-2), the Observer must analyze all live cells in each group at each step – i.e., 5 live cells via $Lens0$, hence 20 cells per 4-step cycle; 8 cells per cycle via $Lens1$ (Cf. Fig. 3); and 6 cells per cycle via $Lens2$. For our 36 step simulation (9 cycles), this results in analyzing 180 cells via $Lens0$, 72 cells via $Lens1$ and 63 cells via $Lens2$. Hence, this resulted in 2.5 reduction factor when using $Lens1$ and 2.8 reduction when using $Lens2$, relative to using $Lens0$.

To detect movement (part of task-2) and identify its trajectory (task-3), the Observer must analyze all plotted centroids (e.g., within a history window for long observations). In our 36 step simulation, this resulted in 19 points for $Lens0$ (Cf. left of Fig. 4); 7 points for $Lens1$ (Cf. right of Fig. 4); and 7 points again for $Lens2$ (not shown). This implies a 5 times reduction factor when changing from a lower lens ($Lens0$) to one of the higher lenses ($Lens1$ or $Lens2$).

Finally, we calculate the angle of the trajectory's linear function (as the arctangent of the x factor, ignoring the sign): 44.7 degrees for $Lens0$; 40.7 degrees for $Lens1$ (hence 4 degree error); and 38.67 degrees for $Lens2$ (hence 6,3 degree error). This gives an error of 0.3degrees (0.6%) for $Lens0$; 4.3degrees (9.5%) for $Lens1$; 6.33degrees (14%) for $Lens2$.

Based on the above analysis, one may conclude that $Lens1$ provides the most suited observations for movement analysis, as only consuming slightly more resources than $Lens2$ while offering better accuracy. Surely, these considerations are relative to each application context.

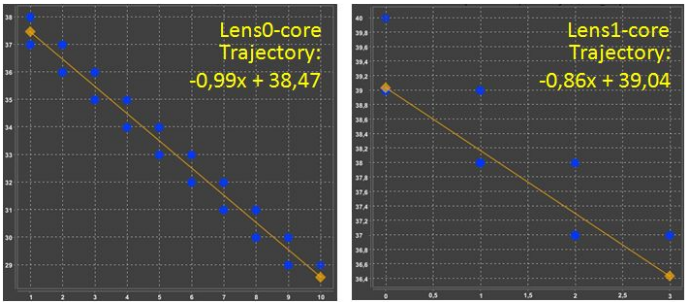


Fig. 4: Centroid plots &trajectory via Lens0-core &Lens1-core

C. Group composition analysis

The following questions require finer-grain observations:

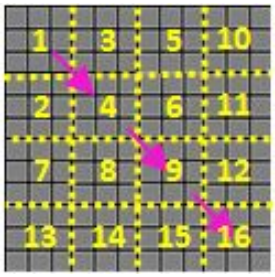
- how many live cells are there in a group? task-4: count the Glider's live cells;
- are the live cells moving within the group? task-5: differentiate between Glider states;

As above, we estimate the efficacy and efficiency of the three lenses for answering the two additional questions above. The higher lenses ($Lens1$ and $Lens2$) are not effective in this case as the questions require cell-level precision; hence lens efficiency does not apply here. Only $Lens0$ can provide the required resolution in this case: the Glider contains 5 live cells, which change position from one step the next (i.e., 4 states).

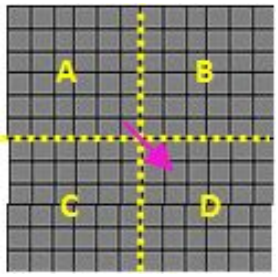
We now ask a question that only requires cell count estimates per patch:

- have any live cells left the observed patch area? task-6: estimate the number of live cells per patch

$Lens0$ can answer this question at every simulation step and irrespective of the number of live cells leaving a patch. $Lens1$ and $Lens2$ can only answer it under certain conditions and only at certain steps. E.g., in the example in Fig. 3, $Lens1$ can only notice cells moving between patches at Step 2 and Step 3; and $Lens2$ only at Step 2. This is because these lenses only provide cell count estimates – i.e., are the number of live cells higher than a threshold? so they can only detect changes that cross this threshold. We can conclude that $Lens1$ and $Lens2$ are somewhat effective at answering task-6; with $Lens1$ being slightly more effective than $Lens2$; and with equal efficiency.



(a) 3x3 patches ($Lens3x3$): the Glider performs 3 diagonal patch transitions during 36 steps (magenta arrows)



(b) 6x6 patches ($Lens6x6$): the Glider performs 1 diagonal patch transition during 36 steps (magenta arrow)

Fig. 5: CA grid of 12x12 cells, at different patch resolutions

IV. Glider Example: Entropy-based Lensing

A. Entropy-oriented Lens

We now reconsider the Glider example from a different perspective, in terms of Observer abilities. We assume that the Observer can only perceive the Cellular Automata (CA) via a lens with maximum resolution of 3x3 cell patches ($Lens3x3$) – Cf. Patches 1 to 16 in Fig. 5a. The Observer also disposes of a lens with lower resolution of 6x6 cell patches ($Lens6x6$) – Cf. Patches A to D in Fig. 5b.

We also assume that the Observer can somehow estimate the total number of live cells within each patch (e.g., the patch color or weight varies with the number of live cells). Still, the Observer cannot establish the exact organization of the live cells (i.e., cannot differentiate among the four glider states).This could be provided by a different kind of lens, not related to the others by resolution differences, that simply provides the number of live cells (the "live weight" of the region).

B. Experimental Settings

We consider a CA grid of 144 cells (12x12), Fig. 5. Depending on the lenses it employs, the Observer perceives a different number of patches N within this grid. Namely, the Observer sees $N = 16$ patches (4x4) when using $Lens3x3$; and 4 patches (2x2) when using $Lens6x6$.

In all cases, we set the initial CA state with the Glider in the upper-left grid corner, as shown in Step 1 of Fig. 2 (i.e., Patch 1 in Fig. 5a and upper-left of Patch A in Fig. 5b). As in Fig. 2, the Glider takes 12 steps to move from one 3x3 patch to the next one diagonally down-right (i.e., from Patch 1 to 4). Considering our 12x12 cells grid, the Glider has to move diagonally between patches for 3 times – i.e., through 4 patches, when using $Lens3x3$ (between patches 1-4, 4-9 and 9-16); and through 2 patches, when using $Lens6x6$ (between patches A and D). Hence, in all cases, we run the Glider for 36 steps (3x12) to have it cross the entire grid.

C. Patch Entropy

At each step t, a patch $i = 1..N$ contains a number of live cells M_i (with $M_i = 0..5$ for the Glider case). M is the total number of live cells in the CA ($M = 5$ at all t for the Glider). There are many different ways to compute entropy analogs, and we have chosen a particularly simple one.

We calculate the entropy analog S_i of each patch i as follows. The measure of occupancy of a patch is $p_i = (1/N)^{M_i}$, and the entropy analog is $S_i = -p_i * log_2(p_i)$. Based on this formula, patches that contain the Glider will have low entropy because they have more live cells, and those that only have part of the Glider (i.e., few live cells) will have high entropy. Fig. 6 lists the number of live cells per patch, the occupancy measure for this and the associated entropy analog for Patches 1 and 4, via $Lens3x3$, during the first 13 steps.

Since the glider is a small part of a patch (less than half for the 3x3 lens and much less for the 6x6 lens, the existence of empty patches is very common. The occupancy measure is 1 by the formula above, so we take $p_{empty} = (1 - 1/N)^M$ instead.

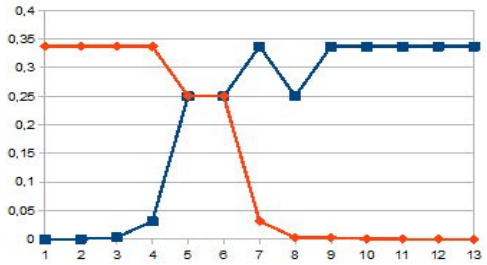
Fig. 7 shows the patch entropy values when the Observer uses the two lenses, over different time periods. The graphs only show values for Patch 1 and 4, as these show the starkest Glider transitions. Patches 2 and 3 also show some disturbances as the Glider passes along their sides (not shown), which could also be interpreted as some sort of movement.

More precisely, Fig. 7a shows the entropy S_4 (of Patch 4, in blue) and entropy S_1 (of Patch 1, in orange), when the Observer uses $Lens3x3$, over the first 13 simulation steps – the Glider moves as shown in Fig. 2 during this time. From then on (steps 13-36, not shown) the entropy of Patch 1 (orange) stays unchanged ($S_{empty} = .33$), as the Glider moves further down. The entropy of Patch 4 changes between steps 13-24 in the same manner as shown in 7a for Patch 1 between steps 1-12 – i.e., changing from lowest to highest entropy as the Glider leaves Patch 4 and goes to Patch 9. After that, the entropy of Patch 4 (blue) remains constant S_{empty} for the rest of the experiment (steps 25-36), as the Glider moves from Patch 9 to 16.

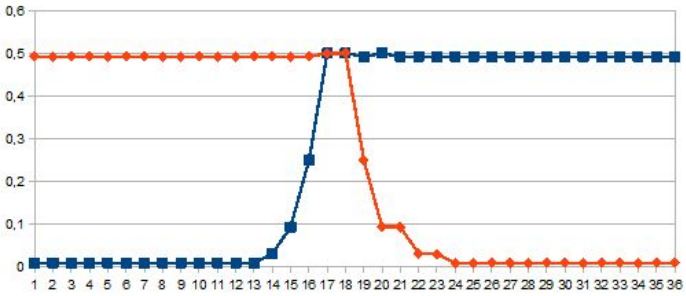
Similarly, Fig. 7a shows the entropy S_D (of Patch D, in blue) and entropy S_A (of Patch A, in orange), when the

Step	Patch1	Patch4	P_Patch1	P_Patch4	S_Patch1	S_Patch4
1	5	0	0,000001	0,724196	0,000019	0,337147
2	4	0	0,000015	0,724196	0,000244	0,337147
3	3	0	0,000244	0,724196	0,002930	0,337147
4	2	0	0,003906	0,724196	0,031250	0,337147
5	1	1	0,062500	0,062500	0,250000	0,250000
6	1	1	0,062500	0,062500	0,250000	0,250000
7	0	2	0,724196	0,003906	0,337147	0,031250
8	1	3	0,062500	0,000244	0,250000	0,002930
9	0	3	0,724196	0,000244	0,337147	0,002930
10	0	4	0,724196	0,000015	0,337147	0,000244
11	0	4	0,724196	0,000015	0,337147	0,000244
12	0	5	0,724196	0,000001	0,337147	0,000019
13	0	5	0,724196	0,000001	0,337147	0,000019

Fig. 6: Entropy values via $Lens3x3$, showing: the simulation Step (grey); Number of live cells for Patch1 & Patch4 (yellow); occupancy measure P_Patch1 & P_Patch4 for that number of live cells to occur in Patch1 & Patch4 (green); and the entropy analog S_Patch1 & S_Patch4 for Patch1 & Patch4 (pink).



(a) Patch entropy values at higher patch resolution ($Lens3x3$); Blue line corresponds to Patch 4; and Orange line to Patch 1.



(b) Patch entropy values at lower patch resolution ($Lens6x6$); Blue line corresponds to Patch D; and Orange line to Patch A.

Fig. 7: Patch entropy via different lenses: values indicate group location and value changes indicate group movement. Both lenses can detect group location; yet $Lens6x6$ requires longer time scope to detect group movement, compared to $Lens3x3$

Observer uses $Lens6x6$, over the entire 36 simulation steps. We note that for the first 13 steps, entropy values stay constant, as the Glider moves across Patch A. A transition is observed between steps 14 and 24 as the Glider transits from Patch A to D. After that, the entropy values stay constant again, as the Gilder transits through Patch D (steps 25-36).

D. Group detection and Movement analysis

We aim to answer the same questions about the Glider's existence, position and movement, as in subsec. III-B.

The entropy values observed through $Lens3x3$ allows the Observer to note that a group exists (low entropy values) and to locate it at the patch level (in Patch 1 during steps 1-4 and in Patch 4 during steps 7-13) – (task-1). The Observer can also note that the group is moving (task-2), as the entropy values change in waves, from one patch to another, during every 12 steps. Finally, the Observer can detect the group's trajectory (task-3), by noting that the group moves diagonally-down every 12 steps (Patch 1-4 during steps 1-12; Patch 4-9 during steps 13-24; and Patch 9-16 during steps 24-36). Hence, $Lens3x3$ is effective for all three tasks.

When using $Lens6x6$, the Observer can also notice the existence of a group (task-1) – e.g., low entropy values for Patch A, during steps 1-15; and for Patch D during steps 20-36. However, the Observer cannot detect the group's movement (task-2) if only observing the CA during steps 1-13, or during steps 24-36. The Observer can only be sure to detect movement when observing the CA at longer time scales (via higher-level time lenses)– e.g., 36 steps as shown in Fig. 7b. Hence, $Lens6x6$ is effective for detecting movement (task-2) if also featuring larger time-scales (not only coarse-grain 6x6 patches). Finally, if observing for sufficiently large time scales to detect group movement, $Lens6x6$ can also determine its diagonal trajectory (e.g., patch A to D) - (task-3).

Neither lens can accurately answer the group composition questions, as they cannot see the group's individual cells (task-4 and 5). However, they can notice when live cells leave an observed patch, via the entropy changes of that patch. hence, they can answer estimate questions about group composition (task-6). Surely, each lens can only answer this question at its own patch resolution.

In summary, both $Lens3x3$ and $Lens6x6$ can be effective in answering the group-detection and movement questions, if observing the CA at sufficiently coarse-grain time scales. Conversely, they are ineffective in answering precise questions on group composition (subsec. III-C), as they cannot see a group's individual live cells. Yet, they can answer these questions approximately. In terms of efficiency, $Lens3x3$ requires more resources than $Lens6x6$ to detect group movement (i.e., having to observe 16 patches rather than 4). At the same time, it requires shorter observation periods than $Lens6x6$ (i.e., 13 steps rather than 36). The best lens here will depend on the Observer's constraints and precision requirements.

V. Conclusion

This position paper aimed to set the stage for a long-term research initiative on CAS *lensing*. The objective is to enable self-aware CAS to determine suitable abstraction levels for observing themselves and other systems, and hence for answering questions or problems as they occur during runtime. Beyond proposing the lensing concept, the paper also introduced associated evaluation notions of *lens effectiveness* and *efficiency*, which will be critical to the self-adaptive lens-selection process. These concepts were illustrated via several examples from collective movement systems, concretely focusing on a Game of Life (GoL) Glider simulation.

Immediate efforts will focus on analyzing further examples where lensing is critical, and proposing a formal methodology for lens selection at runtime – e.g., based on repeated lens acquisition and evaluation cycles. Longer-term efforts will aim to study cases where lenses can be generated automatically, to suit unexpected self-adaptions or to allow for self-integration with other systems, discovered at runtime.

References

[1] J. H. Holland, *Hidden Order: How Adaptation Builds Complexity*. Addison-Wesley, 1995.
[2] H. A. Simon, *The Architecture of Complexity*. Boston, MA: Springer US, 1991, pp. 457–476.
[3] V. Ahl and T. Allen, *Hierarchy Theory*. Columbia Univ. Press, 1996.
[4] P. Benjamin *et al.*, "Simulation modeling at multiple levels of abstraction." 01 1998, pp. 391–398.
[5] M. Gardner, "The fantastic combinations of john conway's new solitaire game "life"," *Scientific American, Mathematical Games*, 10 1970.
[6] S. N. Salthe, *Evolving Hierarchical Systems: Their Structure and Representation*. Columbia University Press, 1985.

[7] J. H. Holland, *Signals and Boundaries: Building Blocks for Complex Adaptive Systems*. MIT Press, 2012.
[8] A. Diaconescu, L. Di Felice, and P. Mellodge, "Exogenous coordination in multi-scale systems: How information flows and timing affect system properties," *Future Gen. Comp. Sys.*, vol. 114, pp. 403–426, 2021.
[9] A. Diaconescu, L. J. Di Felice, and P. Mellodge, "Multi-scale feedbacks for large-scale coordination in self-systems," in *IEEE Intl. Conf. Self-Adaptive and Self-Organizing Systems (SASO)*, 2019, pp. 137–142.
[10] A. Diaconescu, L. Di Felice, and P. Mellodge, "An information-oriented view of multi-scale systems," 09 2021, pp. 154–159.
[11] J. Flack, "Coarse-graining as a downward causation mechanism," *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, vol. 375, p. 20160338, 12 2017.
[12] C. Rovelli, *The Order of Time*. Penguin Publishing Group, 2018.
[13] R. Franceschini *et al.*, "Challenges for automation in adaptive abstraction," in *ACM/IEEE Intl. Cnf. MODELS, Companion, Companion*, L. Burgueño and et.al, Eds., 2019, pp. 443–448.
[14] Y. Iwasaki, "Reasoning with multipl e abstraction models," in *In Faltings, B . and Struss, P . eds . : Recent Advances in Qualitative Physics, Cambridge , Mass*. MIT Press, 1992.