A Data-driven Approach for Modeling Unknown Multi-scale Systems

Marius Pol
Independent researcher
Paris, France
mariuspol@messagebox.email

Ada Diaconescu
LCTI Lab, Télécom Paris, IP Paris,
Palaiseau, France
ada.diaconescu@telecom-paris.fr

***Abstract*—Complex adaptive systems often organize via multiple abstraction levels, or 'scales', interconnected by feedback loops. This enables adaptation and survival in changing environments, while managing complexity with limited resources. For an external observer unaware of such multi-scale structure, modeling an unknown system may be a complicated endeavor. This position paper proposes a data-driven approach for addressing this issue. It generates multi-scale models from incomplete monitoring data, capitalizing on the behavioral regularities that stem from its feedback loops. It also defines the appropriate language elements for expressing these multi-scale models. We validate our approach on data obtained from a theoretical multi-scale system: a holonic cellular automata (HCA) simulator. Results show that the proposed approach can identify the HCA's three abstraction levels and main modeling concepts. This is an encouraging first step towards establishing automatic methods for multi-scale model discovery from partial observations.**

***Index Terms*—knowledge abstraction, multi-scale feedbacks, dynamic meta-models, data-driven modeling**

I. Introduction

Complex systems consist of multiple interacting parts whose collective behavior (higher-level) is difficult to predict based on individual behavior (lower-level). To adapt to changing environments with limited computational resources, these systems organize via multiple abstraction levels, or 'scales' [1], interrelated by feedbacks [2]. E.g. individual trees form forest patches (bottom-up abstraction), which in turn impact further tree growth (top-down feedback). Feedback controllers generally stabilize the dynamics of managed systems [3]. Hence, complex systems organized via multi-scale feedback structures often feature relatively stable high-level behaviors based on more variable low-level dynamics [4]. When modeling such systems, an external observer must select the appropriate scale(s) for their purposes, considering that: coarse-grained models omit details, while detailed models require significant computational power [5]. When the observer is unaware of the system's multi-scale structure, determining the suitable modeling granularity and the key associated concepts (meta-model) represents a considerable challenge.

This position paper proposes a data-driven multi-scale modeling approach to alleviate this issue. It produces multi-scale models from partial monitoring data, capitalizing on the system's behavioral regularities that occur at increasing scales, due to inter-scale feedbacks. Each upper scale is identified as stable behavior based on the lower scale; the lowest scale consists of direct observations. At each scale, we propose to identify key modeling concepts and the associated language (meta-model) at runtime; rather than rely on predefined meta-models, as in 'classic' solutions [6]. We do this by identifying similar reoccurring system states, at each scale, and labeling them as meta-model concepts, at the respective scales.

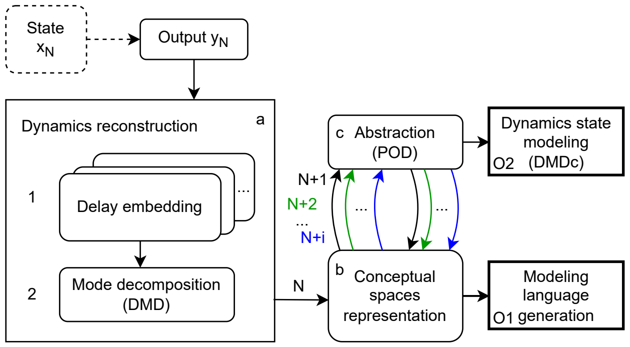


Fig. 1. Overview of proposed approach for *multi-scale modeling*.

Moreover, we propose to infer automatically the system's dynamic state models (i.e. state-transition equations), at each scale. For this purpose, we assume that systems are structured following the *Multi-Scale Abstraction Feedbacks (MSAF)* design pattern [2], [7]. Here, multiple scales affect each other by feedback. The predicted system state $x_N(k+1)$, at scale N and time $k+1$ is computed by merging: i) the system state $x_N(k)$; and ii) the feedback control information from the upper level $N+1$, which we consider to be the abstracted state $x_{N+1}(k)$.

In previous work [8], [9], we proposed a cognitive approach for self-integrating and -improving systems (SISSY) [10], to learn new language elements at runtime, and reason about unknown situations. The approach contains a play module that acquires new knowledge during inactivity periods, so as to improve its reasoning when adapting to new situations. The current proposal extends the learning capabilities of the play module by modeling systems on successive scales, and aims to increase the self-awareness of SISSY systems.

Fig. 1 depicts an overview of our approach. The observation level N is considered the lowest one (i.e. that can be monitored). Monitoring data is given by output y_N, which we assume does not include the full state x_N. To recover the system's main states and dynamics from incomplete measures, we employ a dynamic reconstruction method: *Hankel Dynamic Mode Decomposition (Hankel-DMD)* [11] – block (a) in the figure. This involves applying a 'classic' *Dynamic Mode Decomposition (DMD)* method [12] onto the *time-delay embedded* version of the system's partial monitoring data (input). We then represent the obtained system states into *conceptual spaces* [13] and cluster them (i.e. based on a distance threshold) to obtain key modeling concepts, or meta-types – block (b). These concepts are labeled to extend the meta-modeling language – output (O1). We then apply *Proper Orthogonal Decomposition (POD)* [14] to reduce the dimensionality of conceptual spaces and discover higher-level states x_{N+1} – block (c). Finally, to obtain the dynamic state model at N – output (O2) – we apply *Dynamic Mode Decomposition with Control (DMDc)* [15] on states x_N and control states x_{N+1}.

This process can be repeated recursively, obtaining meta-models (O1) and dynamic state models (O2) at increasingly higher scales. We note that monitoring data is only considered as input at the lowest scale N. Hence, we only need to employ the dynamic state reconstruction (block a) at this scale N. After this, we infer the abstracted states at $N+1$ and use them as input for inferring states at the higher scales.

Generally, we represent the states x_{N+i}, $i > 0$ (obtained via abstraction (c)) into corresponding conceptual spaces. We can then infer the associated modeling concepts (via clustering (b)) and extend the meta-model for scale $N+i$ (O1). These are again abstracted into states x_{N+i+1}, via (c); and again fed back into conceptual spaces to obtain concepts at $N+i+1$. Additionally, states x_{N+i} and upper control states x_{N+i+1} are fed into DMDc to obtain the dynamic model at $N+i$ (O2).

The paper's main contribution is to propose a data-driven approach for dynamic multi-scale modeling from incomplete monitoring data. The approach consists of a sequence of data analysis methods, from the literature, applied recursively to model increasingly higher scales. The process stops when the abstraction process (c) no longer obtains a model reduction.

We validate our approach via a dataset generated by a theoretical multi-scale system: a *Holonic Cellular Automata (HCA)* simulator [16]. Given incomplete monitoring data (lowest scale), our approach generates language elements and determines the dynamic state model parameters that correspond to the observed behavior in the three HCA scales. In future work, we aim to study data from real-world systems with the proposed approach.

II. System Assumptions and Problem Statement

We assume that targeted complex systems fit the following dynamic state equation, at scale N and time k: $x_N(k+1) = f(x_N(k)), k \in \mathbb{N}$, where $x_N(k) \in \mathbb{R}^M$ is the system state, and $f : \mathbb{R}^M \mapsto \mathbb{R}^M$, its nonlinear dynamics. The system equation is obtained by collecting K discrete samples of the state $x_N(k)$ with a rate Δt, i.e. $x_N(k) = x_N(k\Delta t)$, where $k \in \mathbb{N}$. The measured output is provided by a function $g_N : \mathbb{R}^M \mapsto \mathbb{R}^P$

$$y_N(k) = g_N(x_N(k)) \tag{1}$$

where $y_N \in \mathbb{R}^P$. We assume incomplete measurement, with g_N a dimension-reducing function, i.e. $P < M$.

Given the output measurements, our method represents the system's stable repetitive behavior at level N as a linear state space model:

$$x_N(k+1) = A_N x_N(k) + B_N u_N(k) \tag{2}$$

where $u_N(k)$ is the control information at scale N, coming from scale $N+1$, with: $u_N(k) = x_{N+1}(k-1)$.

To generate multi-scale models, our method determines an abstraction transformation T_N for projecting the system state at scale N onto the upper scale $N+1$:

$$x_{N+1}(k) = T_N(x_N(k)) \tag{3}$$

To generate the system's modeling language (O1 in Fig. 1) we must specify its syntax and semantics [17]. The syntax is a set of logic symbols $\mathcal{L}$. A semantic mapping $\mathcal{M}$ provides meaning to syntax elements by relating logic symbols to knowledge representations: $\mathcal{M} : \mathcal{L} \rightarrow \mathcal{C}$. Hence, to model the system at scale N the problem (P1) is to determine the elements in $\mathcal{C}$ to represent the states x_N, and the corresponding logic symbols in $\mathcal{L}$. Furthermore, to generate the system's dynamic state model at scale N the problem (P2) is to determine the values of parameters A_N and B_N in eq. 2.

III. Proposed Approach

This section presents our data-driven method for multi-scale modeling, including language and dynamic model generation at successive scales. The method starts with a dynamics reconstruction step (III-A), which is only required for the observation scale N. Then it determines a subspace for representing states at scale $N+1$ (III-B). Once the state at upper scale $N+1$ is available, it can be used to determine the dynamic model parameters at N (III-C) – addressing (P2). State representation in conceptual spaces allows obtaining the meta-model types for the modeling language (III-D) – addressing (P1). By reapplying the method at the next scale $N+1$, further abstraction levels may be generated.

A. Time-Delay Embedding for Dynamics Reconstruction

We reconstruct the dynamics of the observation level N via the Hankel-DMD method [11], involving two steps: 1) determine a system with the same dynamics as the original, based only on incomplete monitoring data, via a time-delay embedding method [18]; 2) generate the system's stable behavior by computing an optimal basis (i.e. DMD modal decomposition [12]), then compute a state reconstruction based on selected modes (cf. below).

When studying complex systems, we may not have direct access to their entire states, but only to partial measurements via g_N (cf. eq. 1). To reconstruct the state x_N based only on y_N, *time-delay embedding* methods have been proposed for complex nonlinear systems [18]. Delay embedding processes stacked time-shifted versions of y_N, to determine a system equivalent to the original, i.e., for our purposes, it has the same dynamics.

Dynamic Mode Decomposition (DMD) is a data-driven model reduction method for nonlinear systems [12]. DMD extracts an optimal basis from observations, to generate a subspace for a reduced-order representation of a system's dynamics [19]. As system states in multi-scale feedback systems may repeat at different frequencies, or timescales [20], the fact that DMD determines the modes by regrouping distinct frequencies [12] makes it suitable for our approach: modeling the system's stable behavior at the observation level N. The DMD decomposition assumes that full-state measurements are available (i.e. g_N is an identity function, $g_N(k) = k$).

In the first step of Hankel-DMD, the augmented matrix, delayed with a parameter d, and based on K monitoring samples $y_N(k)$, is:

$$Y_N^d(K) = \begin{bmatrix} y_N(1) & y_N(2) & ... & y_N(K-d) \\ y_N(2) & y_N(3) & ... & y_N(K-d+1) \\ & & \vdots & \\ y_N(d) & y_N(d+1) & ... & y_N(K) \end{bmatrix} \quad (4)$$

The system's stable behavior is obtained by modal decomposition of the delayed matrix Y_N^d. The DMD modes are computed supposing the system described by Y_N^d evolves under a linear operator $A_{Y_N}\colon Y_N^d(k+1) = A_{Y_N} Y_N^d(k)$ [21]. The matrix A_{Y_N} is determined as $A_{Y_N} = Y_N^d(k+1)Y_N^d(k)\dagger$, where $\dagger$ is the pseudoinverse – computable for any matrix via Singular Value Decomposition (SVD) (cf. III-B). The dynamics of the delayed matrix are equivalent to the original dynamics, and the state x_N may be reconstructed as a weighted sum of eigenvectors of A_{Y_N}. To select the stable behavior, the weighted sum takes into account only eigenvalues on the unit circle [22].

B. Proper Orthogonal Decomposition for State Abstraction

We determine the next abstraction level $N+1$ by projecting the system state at level N onto a lower dimensional subspace. To determine a basis for this subspace, we use *Proper Orthogonal Decomposition (POD)*, a classic model order reduction method [14]. POD computes an optimal basis for projecting data into a lower order subspace. POD modes may be computed with Singular Value Decomposition (SVD) [23]. SVD decomposes any matrix into the following matrix product:

$$x_N(k+1) = U_N \Sigma_N V_N^T \quad (5)$$

with U_N and V_N orthogonal matrices ($U_N^T U_N = I_{M\times M}$, $V_N^T V_N = I_{K\times K}$), and $\Sigma_N = diag(\sigma_1, \sigma_2, ..., \sigma_r)$, where ^{T} denotes the transpose, σ_i are the singular values, and r the dimension of the lower dimensional subspace. The POD modes are given by the columns of the U_N matrix [23]. We consider the abstraction transformation T_N in equation 3 equal to U_N^T, and obtain the state on the next level of abstraction as $x_{N+1}(k) = \Sigma_N V_N^T$.

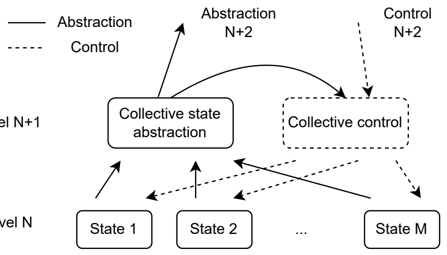


Fig. 2. State abstraction in the MSAF design pattern. States on level N abstract into a collective state on level $N+1$. The collective state is processed to determine the collective control information on level $N+1$, which is used to control states on level N.

C. Dynamic State Model Discovery via MSAF Design Pattern

In the MSAF design pattern the states at level N are abstracted into a collective state at level $N+1$ [7]. Information processing at $N+1$ further computes the collective control, which is transmitted back to level N, leading to state adaptation (figure 2). Each entity at level N merges control information from $N+1$ with its current state information, so as to produce its next state. The state information at level $N+i$, $i>0$, is computed as an abstraction of the states of individual entities at the level below $N-1$.

Determining the system's dynamic state model at level N implies computing parameters A_N and B_N in eq. (2). We follow the MSAF pattern here by considering the control information at level N as the abstracted state computed at level $N+1$, i.e. in the state equation we have $u_N(k) = x_{N+1}(k)$. Given the current and control states at N, the model parameters A_N and B_N may be determined with a DMD variant, *Dynamic Mode Decomposition with control* (DMDc) [15]. DMDc computes the dynamics of linear systems by rearranging the equation 2 in the matrix form as:

$$x_N(k+1) = [A_N B_N] \begin{bmatrix} x_N(k) \\ u_N(k) \end{bmatrix} \quad (6)$$

Similar to DMD, DMDc computes an approximation for the row matrix above, which now contains both A_N and B_N, by multiplying the left side with the pseudoinverse of the column matrix in 6, which includes both the state and the input.

D. Knowledge Representation in Conceptual Spaces

To generate the language elements for modeling system states at level N, we represent the states in equation 2 into *conceptual spaces* [13]– a cognitive knowledge representation framework where data is represented as vectors in geometrical spaces. These vectors denote the centroids of concepts, i.e. convex regions regrouping similar observations. Conceptual spaces are built on domains, which are defined over one or several dimensions relevant for high-level goals. Our proposal generates, as successive levels of abstraction, concepts modeling the behavior to achieve these goals, assumed unknown.

In this multi-dimensional geometric space, a distance measure may be defined to provide a similarity measure between state observations. Here, we consider the euclidean metric as a

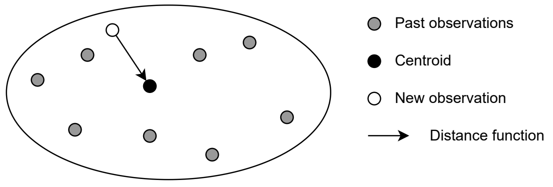


Fig. 3. Conceptual spaces representation. Similar past observations are aggregated into a single concept given by the centroid of the region enclosing them. Concept membership for new observations is computed as a similarity measure, given by their distance to the centroid.

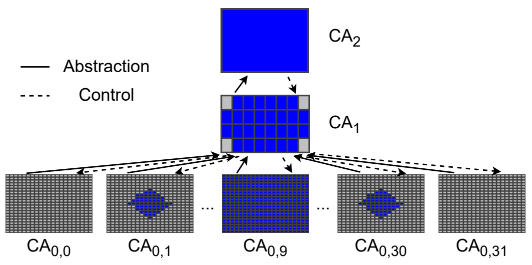


Fig. 4. HCA simulator with three abstraction levels

distance function though other metrics may also be used [13]. Based on the similarity measure, we classify new observations via a function that assigns a degree of membership to existing concepts $c \in \mathcal{C}$ [24]. The function takes values in the interval [0, 1]. A state observation is classified into a concept for which the degree of membership is greater than a threshold Th_{dm}.

Classifying new observations, when the set of known concepts $\mathcal{C}$ is not empty, is shown in figure 3. The black dot is the centroid of a concept c generated by past observations (gray dots). Concept membership is given by the degree of membership of the new observation (white dot) to the centroid being greater than Th_{dm}. When a new observation cannot be classified into a known concept, a new concept is created and added to the set $\mathcal{C}$.

Our method adds conceptual representations to $\mathcal{C}$, and then, via a static labeling function, generates unique symbols for these concepts. Symbols are added to $\mathcal{L}$ as new language elements. The correspondence between concepts and symbols provides the semantic mapping $\mathcal{M}$. In the conceptual spaces, any observation close enough to the centroid of a concept $c \in \mathcal{C}$ is classified as c.

IV. Experiments and Results

We apply the proposed approach for automatically modeling a Holonic Cellular Automata (HCA) [16] with three scales.

A. Experimental Setup

The HCA simulator exemplifies a theoretical multi-scale feedback system, with three abstraction levels. The bottom level 0 contains 32 CAs ($CA_{0,i}$, $i = 0...31$ in the simulation); the middle level contains one CA_1 (with 32 cells); and the top level one CA_2 (with one cell) – fig. 4. Each CA consists of a grid of cells, each with two possible states: $Alive$ or $Dead$ (e.g. CA_1 in fig. 5-Left). Cell states change at each step as dictated by a set of active rules. Each CA holds two sets of rules (cf. [16]), one of which is activated via a control state. The active rule of the top CA_2 is constant and switches its cell state at each cycle. Each of the 32 $CA_{0,i}$ is mapped to one of the 32 cells in CA_1; the entire CA_1 is mapped to the one cell of CA_2. This cross-level mapping means that the entire state of a $CA_{0,i}$ is abstracted into the state of a cell in CA_1; and, the state of each CA_1 cell sets the active rules for a $CA_{0,i}$. The same holds between CA_1 and CA_2.

Bottom-up state abstraction is defined as: if the number of $Alive$ cells at $CA_{0,i}$ exceeds a threshold Th_0, the state of the mapped cell in CA_1 is $Alive$; otherwise it is $Dead$. The same applies for abstracting the CA_1 state into CA_2's cell state, using a threshold Th_1. At each simulation cycle, all CAs at each level: execute their active rule; send their abstracted state upwards; receive the control state from above. The delay between the execution cycles of the three levels is given by their activation frequencies (Fq_0, Fq_1, Fq_2).

The results presented here are for a dataset (available online[1]) generated by the HCA with the following parameters: thresholds Th_0=0.1 and Th_1=0.7; activation frequencies Fq_0=1, Fq_1=1 and Fq_2=3; each CA_0, i has 21x21=441 cells.

B. Results in the HCA Simulator

We implemented the proposed method (cf. III) in Python. We provided 32 measurements as input: the number of live cells of each $CA_{0,i}, i = 0..31$ – obtained via a monitoring function g_0 (eq. 1). The 32 dimensions corresponding to measurements of g_0 are denoted as $L0D00$ to $L0D31$. We validate our approach by modeling the states of CAs at all three levels, based on g_0, and comparing the results with the ones available in the HCA simulation dataset.

The simulated behavior of CA_1, once stabilized, cycles (6 steps) through 2 distinct states (figure 5). We refer to these as Core (12 center cells) and Cross (12 center cells plus 16 border cells). The behavior at $CA_{0,i}$ stabilizes into cycles (6 steps) of 4 states (IDs=4..7). CA_2 cycles through 2 states (in 6 steps). Next, we compare these simulated behaviors (expected) with the ones obtained via our method (modeled). Monitoring data obtained via g_0 contains K=170 samples, each of P=32 measurements, forming a dataset of 32 rows by 170 columns. To determine the stable modes at this level 0, we use an existing Hankel-DMD implementation: PyDMD [25][2]. The delay parameter is $d = 2K/P$ [22] (to compute more accurately in future work [26]).

To reconstruct the state on level 0, we use the 'first' method available for Hankel-DMD in PyDMD. This returns an approximation of the $Y_0^d(k)$ matrix where only stable modes have been selected. The 'first' method corresponds to the first version of the state in the delayed matrix (as in [27]). The reconstruction at level 0 identifies 4 states (i.e. 4 value groups (86.70), (52.18, 62.22), (41.97), (113.35)), which correspond to the 4 expected states from the HCA simulator (respectively,

[1]https://github.com/adadiaconescu/hca/wiki, last access July 2023
[2]https://github.com/mathLab/PyDMD, last access July 2023

TABLE I
RESULTS IN THE HCA SIMULATOR ON LEVEL 0

	Time step	1	2	3	4	5	6
Expected	State Type	6	5	4	5	6	7
	Live Cells	85	61	41	61	85	113
Generated	L0D01	86.70	52.18	41.97	62.22	86.31	113.35

with 85, 61, 41, 113 live cells) – as in Table I. The table presents the results for 6 time steps (1 cycle), with the 4 detected states that repeat at each cycle (Type IDs: 4,5,6,7). The expected state types and live cells counts are available in the simulator dataset. The results shown are for $CA_{0,1}$, which is mapped to a border cell in CA_1 (i.e. the second cell in the first row in Fig. 4 and 5).

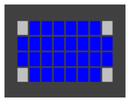
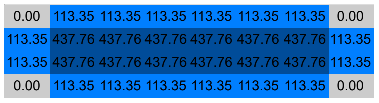
Once we detected the stable system states, we can represent them within conceptual spaces. The dimension of reconstructed states is smaller than that of the measured states, which should also be smaller than the original [18]. At HCA level 0 we have an original state of 441 cells and a measurement dimension of P=32. The dimension for the reconstructed state for level 0 is R=32, just as the one of the measurement state $P = 32$. Since generally $R < P < M$, we consider R as a sufficient dimension reduction of the original state ($M = 441$), and generate the conceptual space based on the reconstructed state.

For knowledge representation in the proposed approach, we rely on an existing implementation of the conceptual spaces framework [24] [3]. We consider all available dimensions to construct conceptual spaces for unknown high-level goals. Clustering relevant dimensions that generate spaces for diferent high-level goals is subject of future work.

The proposed method generates concepts for the values of the reconstructed state on dimensions $L0D00$ to $L0D31$. To generate a concept, we compute the starting and ending values of the region on the associated dimension by setting a threshold Th_{gen} relative to the state value of the concept to be represented. For the value of the state on dimension m, $x_{0,m}(k)$, the boundary of the segment is [$x_{0,m}(k)$ * (1 - Th_{gen}), $x_{0,m}(k)$ * (1 + Th_{gen})]. The value of Th_{gen} determines the concept resolution, hence influencing the number of generated concepts. When generating concepts, our method determines if states observations belong to known concepts by comparing the value of the degree membership function to the threshold Th_{dm}=0.9 for known concepts. Observations falling into regions denoting existent concepts are represented as these concepts. New observations on a dimension generate concepts based on Th_{gen}.

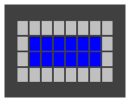
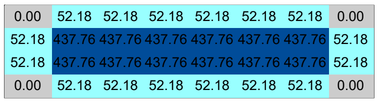
The expected states of CA_1 in the simulator are shown in figure 5, left: Cross state (a) and Core state (b). Our approach generates two corresponding multi-dimensional concepts, representing the states of $CA_{0,i}$ that are mapped to cell states in CA_1 – figure 5, right. Here, each cell corresponds to a cell in CA_1 mapped to a $CA_{0,i}$; and each value is the centroid of the generated concept. On the dimension $L0D01$

[3]https://github.com/lbechberger/ConceptualSpaces, last access July 2023

0.00	113.35	113.35	113.35	113.35	113.35	113.35	0.00
113.35	437.76	437.76	437.76	437.76	437.76	437.76	113.35
113.35	437.76	437.76	437.76	437.76	437.76	437.76	113.35
0.00	113.35	113.35	113.35	113.35	113.35	113.35	0.00

a) Cross

0.00	52.18	52.18	52.18	52.18	52.18	52.18	0.00
52.18	437.76	437.76	437.76	437.76	437.76	437.76	52.18
52.18	437.76	437.76	437.76	437.76	437.76	437.76	52.18
0.00	52.18	52.18	52.18	52.18	52.18	52.18	0.00

b) Core

Fig. 5. Expected and generated *Cross* and *Core* patterns (live cells are in blue). Left: expected states of CA_1, based on simulation results . Right: generated concepts to represent the CAs on level 0.

TABLE II
RESULTS ON LEVEL 1 IN THE HCA SIMULATOR.

	Time step	1	2	3	4	5	6
Expected	L1	Cross	Cross	Core	Core	Core	Cross
Generated	L1D0	-1,575	-1,575	-1,523	-1,523	-1,523	-1,575
	L1D1	-115	-115	122	122	122	-115

(the second cell in the first row), the approach generates a concept with centroid at 113.35 (which we label “Alive”) and another concept with centroid at 52.18 (labeled “Dead”). These two concepts correspond to the live and dead states observed in the simulation for the mapped cells at CA_1, respectively. Similar concepts are generated for every dimension ($L0D00$ to $L0D31$), using Th_{gen}=0.3. Concepts are only generated based on data monitored after system stabilization (i.e. cyclical behavior); data from the start of the simulation is excluded.

To generate the state at level 1, we project the state at level 0 onto a subspace obtained via SVD decomposition, available in PyDMD [25]. The SVD operation is used with an optimal rank truncation, keeping only the most important modes associated to the singular values determined by the decomposition.

The results obtained at level 1 are shown in table II. The projection from the space at level 0 to the space at level 1 results in a reduction of space dimensions from 32 to 2. The generated values on these two dimensions at level 1 correspond to the $Cross$ and $Core$ patterns observed in the simulator. The table contains only the integer part of the obtained values. We note that the values obtained on the two dimensions allow to differentiate two states at level 1 – e.g. value pairs (-1575;-115) and (-1523;122) correspond to the Cross and Core states, respectively. The two concepts are added to the knowledge set $\mathcal{C}$ and labeled as $\{Cross, Core\}$. These two labels designate the language elements which are added to $\mathcal{L}$. The semantic mapping $\mathcal{M}$ is updated with the correspondence between the new concepts and their associated language elements.

Finally, we determine the parameters of the dynamic state equation for level 0 – A_0 and B_0 – with DMDc (also available in PyDMD). We can now predict the state at level 0 as in equation 2. We similarly determine parameters A_1, B_1 for level 1. We note that B_1 an inverted identity matrix, which does indeed model the expected control behavior from level 2, based on a state inversion rule.

V. Related Work

Traditional modeling approaches focus on specific scales of interest. To solve the difficulties of determining these scales multi-scale modeling methods have been proposed [5]. Our proposal here is to determine automatically all modeling scales that may be of interest to an external observer.

For determining the scales of interest, a possible approach is to separate fast and slow time scales (e.g. [28]). This differs from our proposal in that we propose to identify multiple time scales, associated to the DMD modes which regroup distinct frequencies (Cf. III-A). In the Complex Automata Model [29], the levels of abstraction are determined based on a scale separation map on two dimensions, space and time. The scale separation map determines the different levels of abstraction and interactions between them. However, this is expressed in a fixed modeling language.

Complex systems modeling methods focus on representing their composing parts and the relationships between them. The representations for modeling developed, for example, from elementary cellular automata [30], to networks modeled as graphs [31], by generalizing the relationship type. Hypergraphs [32] generalize graphs by supporting multiple relationships between parts, and are closer to the knowledge representation framework of conceptual spaces in out proposal.

VI. Conclusions and Future Work

We proposed a data-driven approach for automatically generating multi-scale models for complex adaptive systems, based on incomplete monitoring data. This includes both: i) defining the multi-scale modeling language (meta-models); and ii) determining the model state dynamics at each scale. Our method involves several data-processing techniques from the literature, notably: Hankel-DMD for recovering system state from incomplete monitoring data; conceptual space representations for concept identification and language generation; POD for state abstractions and new scale identification; and DMDc for dynamic state modeling. We validated the proposed approach via a holonic CA simulator with three scales. Results showed that our method managed to generate models at the three different scales and that these models corresponded to the states and dynamics observed in the simulation. This is an encouraging first step towards automatic methods for multi-scale model discovery from system observations. In future work we will apply the proposed method to more realistic case studies; and see how automatic reasoning can be performed on generated knowledge.

References

[1] H. A. Simon, "The Architecture of Complexity," *Proc. Am. Philos. Soc.*, vol. 106, no. 6, pp. 467–482, 1962.

[2] A. Diaconescu, L. J. Di Felice, and P. Mellodge, "Multi-Scale Feedbacks for Large-Scale Coordination in Self-Systems," in *IEEE Int. Cnf. Self-Adapt. Self-Organ. Syst SASO*, Umea, Jun. 2019, pp. 137–142.

[3] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. Hoboken, NJ: John Wiley, 2005.

[4] J. Flack, D. Erwin, T. Elliot, and D. Krakauer, "Timescales, symmetry, and uncertainty reduction in the origins of hierarchy in biological systems," *Evolution, cooperation and complexity*, pp. 45–74, 2013.

[5] E. Weinan, *Principles of Multiscale Modeling*. Cambridge ; New York: Cambridge University Press, 2011.

[6] G. Lehmann, M. Blumendorf, F. Trollmann, and S. Albayrak, "Meta-modeling Runtime Models," in *Models in Software Engineering*, J. Dingel and A. Solberg, Eds. Springer, 2011, vol. 6627, pp. 209–223.

[7] A. Diaconescu, L. Di Felice, and P. Mellodge, "Exogenous coordination in multi-scale systems: How information flows and timing affect system properties," *FGCS*, vol. 114, pp. 403–426, 2021.

[8] M. Pol and A. Diaconescu, "A Cognitive Control System for Managing Runtime Uncertainty in Self-Integrating Autonomic Systems," in *IEEE Intl. Cnf. ACSOS*, 2020, p. 6.

[9] ——, "Multi - Level Online Learning and Reasoning for Self-Integrating Systems," in *2021 IEEE Int. Conf. Auton. Comput. Self-Organ. Syst. Companion ACSOS-C*. DC, USA: IEEE, Sep. 2021, pp. 187–192.

[10] K. Bellman, J. Botev, A. Diaconescu, L. Esterle, C. Gruhl, C. Landauer, P. R. Lewis, P. R. Nelson, E. Pournaras, A. Stein, and S. Tomforde, "Self-improving system integration: Mastering continuous change," *Future Generation Computer Systems*, vol. 117, pp. 29–46, Apr. 2021.

[11] H. Arbabi and I. Mezić, "Ergodic theory, Dynamic Mode Decomposition and Computation of Spectral Properties of the Koopman operator," *SIAM J. Appl. Dyn. Syst.*, vol. 16, no. 4, pp. 2096–2126, Jan. 2017.

[12] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *J. Fluid Mech.*, vol. 656, pp. 5–28, Aug. 2010.

[13] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought*, 2000.

[14] R. Pinnau, "Model Reduction via Proper Orthogonal Decomposition," in *Model Order Reduction: Theory, Research Aspects and Applications*. Springer Berlin Heidelberg, 2008, vol. 13, pp. 95–109.

[15] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," Sep. 2014.

[16] A. Diaconescu, S. Tomforde, and C. Müller-Schloer, "Holonic Cellular Automata: Modelling Multi-level Self-organisation of Structure and Behaviour," in *Artif. Life*. Tokyo, Japan: MIT Press, 2018, pp. 186–193.

[17] D. Harel and B. Rumpe, "Modeling Languages: Syntax, Semantics and All That Stuff Part I," *URL Httpwww Ncstrl Org*, 2000.

[18] M. Casdagli, S. Eubank, J. Farmer, and J. Gibson, "State space reconstruction in the presence of noise," *Physica D: Nonlinear Phenomena*, vol. 51, no. 1-3, pp. 52–98, Aug. 1991.

[19] G. Tissot, L. Cordier, N. Benard, and B. R. Noack, "Model reduction using Dynamic Mode Decomposition," *Comptes Rendus Mécanique*, vol. 342, no. 6-7, pp. 410–416, Jun. 2014.

[20] P. Mellodge, A. Diaconescu, and L. Di Felice, "Timing configurations affect the macro-properties of multi-scale feedback systems," in *IEEE Intl. Cnf. ACSOS*, 2021, pp. 100–109.

[21] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Soc. for Industrial and Applied Mathematics, Nov. 2016.

[22] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz, "Extracting spatial-temporal coherent patterns in large-scale neural recordings using ddm," *J. Neuroscience Methods*, vol. 258, pp. 1–15, 2016.

[23] P. Holmes and P. Holmes, Eds., *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, 2012.

[24] L. Bechberger and K.-U. Kuhnberger, "A Comprehensive Implementation of Conceptual Spaces," *ArXiv Prepr. ArXiv170705165*, p. 14, 2017.

[25] N. Demo, M. Tezzele, and G. Rozza, "PyDMD: Python Dynamic Mode Decomposition," *JOSS*, vol. 3, no. 22, p. 530, Feb. 2018.

[26] E. Tan, S. Algar, D. Corrêa, M. Small, T. Stemler, and D. Walker, "Selecting embedding delays: An overview of embedding techniques and a new method using persistent homology," *Chaos*, p. 032101, 2023.

[27] D. Dylewsky, E. Kaiser, S. L. Brunton, and J. N. Kutz, "Principal component trajectories for modeling spectrally-continuous dynamics as forced linear systems," Jan. 2022.

[28] K. Champion, S. L. Brunton, and J. N. Kutz, "Discovery of Nonlinear Multiscale Systems: Sampling Strategies and Embeddings," *SIAM J. Appl. Dyn. Syst.*, vol. 18, no. 1, pp. 312–333, Jan. 2019.

[29] A. G. Hoekstra, E. Lorenz, J.-L. Falcone, and B. Chopard, "Towards a Complex Automata Framework for Multi-scale Modeling," in *Computational Science – ICCS 2007*. Springer, 2007, vol. 4487, pp. 922–930.

[30] S. Wolfram, "Cellular automata as models of complexity," *Nature*, vol. 311, no. 5985, pp. 419–424, Oct. 1984.

[31] N. Boccara, *Modeling Complex Systems*, ser. Graduate Texts in Physics. New York, NY: Springer New York, 2010.

[32] A. Bretto, *Hypergraph Theory: An Introduction*, ser. Mathematical Engineering. Heidelberg: Springer International Publishing, 2013.