What should I notice? Evaluating the memorability of events for abductive inference.

*† Étienne Houzé, * Jean-Louis Dessalles, * Ada Diaconescu, † David Menga
* Télécom Paris, IP Paris, *Palaiseau*, France
email: {first}.{second}@telecom-paris.fr
† EDF R&D, *Palaiseau*, France
email: {first}.{second}@edf.fr

***Abstract*—When confronted to an unprecedented situation, humans typically show good performance in quickly identifying noticeable past events and proposing them as possible causal hypotheses. This kind of abductive inference is widely overlooked in modern AI approaches which rely on massive datasets to learn associative patterns. Our proposal is to formalize and compute a "memorability" score over a set of events recorded from a cyber-physical system. This score can then be used to select relevant information to be remembered and to propose causal hypotheses in unusual situations on demand. The approach is meant to be complementary to more traditional learning-focused abduction techniques. We provide theoretical ground for our approach based on Algorithmic Information Theory. We also provide an implementation example, a smart-home scenario, to show how our approach could translate in practice.**

***Index Terms*—Kolmogorov complexity, Algorithmic Information Theory, Simplicity, Abduction, Surprise**

I. Introduction

As a user has just turned on the TV in her all-equipped living-room, the lights dim and the window blinds go down. Intrigued by this behavior, she quickly infers that both light dimming and blind closing occurred as a consequence of the TV being turned on. How did she come to this conclusion? By performing *abductive inference* [1]. This mental operation is a key element of humans' ability to understand the world: from the observed consequences, they infer the possible causes.

In this example, there are mainly three possibilities to come to the conclusion. (1) If the user knows how the smart living-room system works, if she knows the underlying rules or parameters, she may use this causal knowledge to perform abduction. (2) If she has no knowledge about the system but made several observation of the same behavior, she may examine past correlations and figure out that turning on the TV set often leads the blinds to close and the lights to dim. (3) If there are no previous occurrences of the event (e.g. it is the first time she turns on the TV in the living-room), she may still be able to suspect that the TV is a possible cause for the observed event, just because it appears to her as a memorable recent event (as it is its first occurrence). This example suggests that human beings are able to use distinct methods to perform abductive tasks and infer new knowledge. While the first two mechanisms can be automated using knowledge bases and statistical methods, "memorability-based" abduction needs to be investigated in more depth.

Defining a score of memorability is not straightforward. First, events can be of different nature, and not directly comparable. Even for restricted systems such as smart homes, noticeable events range from device removal to presence detection or unusually high temperatures. Even for comparable events, the problem is to weigh different characteristics: is a record-high temperature 47 days ago more memorable than the small deviation recorded just 3 minutes ago? To our knowledge, no current system proposes to combine various event types from different devices to compute a unified metrics of "memorability".

To address this issue, we started from the following intuition: while all events, regardless of their characteristics or nature, can be uniquely described using a combination of quantitative or qualitative qualifiers, the most memorable ones are likely to require less words to be described. "Last year's hottest day" is simpler than "the 182$^{\text{nd}}$ day 7 years ago". How can we quantify this relative simplicity? We could evaluate the complexity of each description, taking into account both the complexity of the concept words (a date of occurrence, a temperature ranking), and of the arguments (1st hottest, 182 and 7). This approach, The resulting values define the *description complexity* of events. Our intuition is that memorable events require simpler and less numerous qualifiers to be unambiguously described than unremarkable ones.

For machines to implement and compute description complexity, we need a formal framework and computation methods that are in line with human intuition. Algorithmic Information Theory (AIT) appears to be such a framework, as it is consistent with the human perception of complexity [2]–[4]. Our method is as follows: we consider events as being elements stored in what we call *base memory*. To reproduce the language features applicable to events, we use *predicates*, i.e. functions assigning a boolean value to events. For instance, the predicate $\texttt{date}(\cdot, 1\_year)$ is *true* of events that occurred last year. Selecting all events, from the memory, that satisfy a given predicate corresponds to a *filter* operation. It generates another memory that is a subset of the previous one. The filtering operation can then be repeated, selecting fewer events at each iteration, until a singleton memory is reached. This means that the sequence of predicates could unambiguously *retrieve* the unique remaining event. The description complexity of this event can thus be upper-bounded by the number of bits

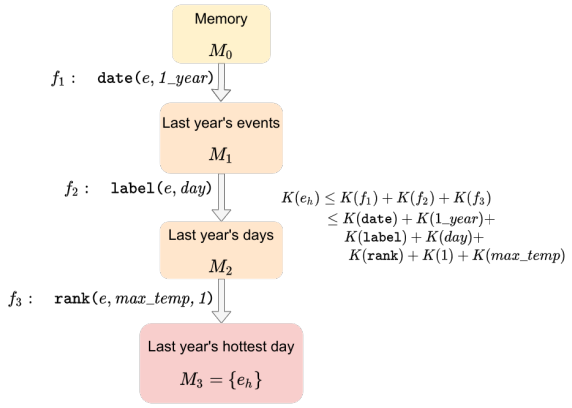

Fig. 1. Retrieving an event through successive predicative filters. From the base memory (yellow), successive filters select events satisfying the associated predicate (grey arrows). For example, filter f_1 selects events from last year, i.e. which satisfy the predicate $\texttt{date}(event, 1\_year)$. In this case, successively applying filters f_1, f_2 and f_3 yields a unique event e_h, last year's hottest day. The complexity of this event can then be upper-bounded by the complexity of the three filters, since they give an unambiguous way to describe the event within the memory.

required to describe the filters used in the retrieval process. Figure 1 illustrates this process for the event: "last year's hottest day".

We will first briefly introduce some basic notions of Algorithmic Information Theory. Then, in section III, we show how we compute the memorability of past events. We illustrate our approach in (section IV) by describing an implementation in a smart home simulation context. In section V we review a few studies that use complexity theory to generate explanations for the smart home and we discuss how our work can be linked with them. We eventually explore potential extensions of our work in section VI

II. Theoretical Background

Kolmogorov complexity formally quantifies the amount of information required for the computation of a finite binary string[1] (or any object represented by a finite binary string) [2], [5]. The complexity $K(s)$ of a (finite) binary string is the length in bits of the shortest program p which, if given as input to a universal Turing Machine U, outputs s.

$$K_U(s) = \min_p \{l(p) | U(p) = s\} \tag{1}$$

The first notable property of this definition is its universality: while the choice of the Turing machine U used for the computations appears in the definition of eq. 1, all results hold, up to an additional constant, if we change the machine. Think how any Turing-complete programming language can be turned into any other language, using an interpreter or a compiler program. Since any Turing machine U' can be emulated by U from a finite program p_U, we have the following inequality:

$$K_{U'}(s) \leq l(p_u) + K_U(s) \tag{2}$$

[1]Though the definition holds for some infinite binary strings (think of the representation of the decimals of π), we restrict ourselves here to finite strings.

From this first result, we can then define complexity $K(s)$, based on the choice of a reference Turing machine, such that, for any other machine U taken from the set TM of Turing machines:

$$\forall U \in \text{TM}, \forall s, |K(s) - K_u(s)| \leq C_U \tag{3}$$

where the additional constant C_U does not depend on s.

Note that the notion of Kolmogorov complexity involves no requirement on the execution time of programs, only their length in bits matters for the computation of complexity. Though Kolmogorov complexity can be shown to be incomputable [2], it can be approximated with upper bounds by exhibiting a program outputting s.

Interestingly, Kolmogorov complexity matches the intuitive notion and perception of complexity from a human standpoint. For instance, the complexity of short binary strings evaluated in [4] shows similar results to human perception of complex strings and patterns. More recently, [6] used Kolmogorov complexity to solve analogies and showed results close to human expectations.

The bridge between Algorithmic Information Theory (AIT) and human perception of complexity can be pushed farther thanks to the notions of simplicity and unexpectedness, which are sometimes regarded of uttermost importance in cognitive science [7]. [3] proposes a formal definition of the unexpectedness $U(e)$ of an event, as the difference between an a-priori expected causal complexity $K_w(e)$ and the actual observed complexity $K(e)$.

$$U(e) = K_w(e) - K(e) \tag{4}$$

This result comes from the understanding that, while Kolmogorov complexity is ideally computed using a Turing machine, it can be used as a proxy for modeling information processing in the human brain, and thus can be used to define a notion of simplicity or complexity of events.

Definition 4 allows to model phenomena such as coincidences: imagine that you happen to run into someone in a park. If this person has no particular link to you, the event will be quite trivial: the complexity of describing this person will be equivalent to distinguishing it from the global population, which is also roughly equivalent to the (causal) complexity of describing the circumstances having brought this person to be in that park as the same time as you. On the other hand, if you run into your best friend in a park, as the complexity of describing your best friend is significantly lower, the description complexity $K(e)$ drops while the causal complexity $K_W(e)$ remains unchanged. This is why this latter event appears unexpected.

Using these insights from AIT, we define the memorability $M(e)$ of an event as the absolute difference between the description complexity $K_d(e)$ of an event and its expected description complexity $K_{exp}(e)$:

$$M(e) = |K_{exp}(e) - K_d(e)| \tag{5}$$

Contrary to the definition of unexpectedness from Equation 4, we use an absolute value: we do so to acknowledge the fact

that events more complex than expected can be memorable as well[2]. In the next section, we define computational approximations for the description complexity K_d and the expected complexity K_{exp} of events.

III. Computing the memorability of events

A. Retrieving an event

We define *events* as data points augmented with a *label* indicating their nature (temperature event, failure event, addition/removal of a device) and a timestamp of occurrence. Formally:

$$e = (l, t, \mathcal{D}) \tag{6}$$

where l is the label, t the timestamp and $\mathcal{D}$ a multi-dimensional data point representing the various characteristics of e: its duration, the maximum temperature reached, the sensor name, its position, etc. Labels can also be considered as classes of events, of which each event is a particular instance.

To model how humans are able to describe events by using qualifiers, we use *predicates*: Boolean functions operating on events and, possibly, additional parameters: $\pi(e, a_1, a_2, \ldots, a_n) \mapsto \{O, 1\}$ is a predicate of arity n operating on event e. In the rest of this paper, we will prefer the equivalent notation $\pi(e, k) \mapsto \{0, 1\}$, where k is a binary string encoding the sequence of arguments $a_1, \ldots, a_n$. Using this notation, the predicate π becomes a boolean function operating on $\mathbf{E} \times \{0, 1\}^*$:

$$\pi : \begin{cases} \mathcal{M} \times \{0,1\}^* & \mapsto \{0,1\} \\ (e, k) & \mapsto \pi_k(e) \end{cases} \tag{7}$$

As an example of predicate, consider $\pi = \texttt{year}$ and k a string encoding the number 1, thus constructing the predicate $\texttt{year}(e, 1)$, which tells whether the event e occurred 1 year ago.

As events occur, they are stored in the *base memory* M_0. As they are not directly comparable, the memory M_0 can be considered as having the structure of an unordered set. We denote by $\mathcal{M}$ the subsets of M_0. By extension, elements of $\mathcal{M}$, i.e. subsets of M_0, are also called *memories*.

By applying a given predicate π to all events contained in a memory $M \subseteq M_0$, and selecting only events satisfying π, one gets another memory $M_1 \subseteq M \subseteq M_0$. We call this operation a *filter*:

$$f_{\pi,k} : \begin{cases} \mathcal{M} & \mapsto \mathcal{M} \\ M & \mapsto \{e \in M | \pi(e)\} \end{cases} \tag{8}$$

For instance, using the same $\pi = \texttt{year}$ and $k = 1$ as above, we can build the filter $f_{\pi,k} = \texttt{last\_year}$, which selects all events that occurred last year.

As the output of a filter applied to a memory M is another memory object $M' \subseteq M$, we can compose filter functions. A sequence of such filters is called a *retrieval path*

$$p = (f_{\pi_1,k_1}, \ldots, f_{\pi_n,k_n}) \tag{9}$$

[2]In the original paper [3], exceptionally complex events are described by considering complexity itself as a way to describe the event: see "the Pisa Tower effect" [8]

By definition $p(M) = f_{\pi_n,k_n}(\ldots(f_{\pi_1,k_1}(M)))$. In case the result of the operation $p(M)$ contains a single element e, we say that the path p *retrieves* the element e from M, and write $p(M) = e$. In the example shown in Figure 1, the three filters f_1, f_2, f_3 form a retrieval path retrieving the event "last year's hottest day" from the base memory M_0.

B. Description complexity of events

As presented in sec. II, we are interested in computing an approximation of the description complexity of an event e. From the above definitions, if there is a path p retrieving e from the base memory M_0, i.e. $p(M_0) = e$, this path provides a possible unambiguous description for e. We therefore define the description complexity of e as the minimum complexity of a path p retrieving e from the base memory M_0.

$$K_d(e) = \min_{p \in P_\infty} \{L(p) | p(M_0) = e\} \tag{10}$$

where the bit-length $L(p)$ of a retrieval path is defined as the number of bits of a string encoding the path. If we limit ourselves to prefix-free strings encoding predicates and arguments, the total bit length is given by:

$$L(p) = L((f_{\pi_1,k_1}, \ldots, f_{\pi_n,k_n})) \tag{11}$$

$$= L(\pi_1) + L(k_1) + \cdots + L(\pi_n) + L(k_n) \tag{12}$$

By considering only a finite number of possible predicates π and arguments k, and a maximum path length, we can construct a finite set P of possible retrieval paths. By limiting the search over this set, we get an upper bound of description complexity, and use this upper bound as an approximation:

$$K_d(e) \leq \min_{p \in P \wedge p(M_0) = e} L(p) = \min_{p \in P \wedge p(M_0) = e} \sum_{f_{\pi,k} \in p} L(\pi) + L(k) \tag{13}$$

The approximation of description complexity from Equation 13 allows for a direct implementation, which is shown in Algorithm 1. This algorithm operates iteratively: starting from the base memory M_0 (line 1), we apply all possible predicate concepts π from a given finite set Π and programs k (lines 6-7), up to a given length `max_len` bits, and apply them: $M' = f_{\pi,k}(M)$ (line 12). We then store the pairs $(M', \texttt{len}(\pi, k))$ in an array $\texttt{future}_{\texttt{explore}}$. At the end of the iteration, the results of the filters become the memories which will be explored during the next iteration(lines 21–23). Each pass thus explore retrieval paths of increasing length. When a singleton memory is reached, the complexity of its unique element is upper-bounded with the length of the corresponding retrieval path (line 14).

C. Computing Memorability

As stated in Equation 5, we define memorability $M(e)$ as the absolute difference between the description complexity of an event and its expected value. As we've just defined $K_d(e)$ and provided an approximation in Equation 13, we now focus on defining the *expected* description complexity of an event, $K_{exp}(e)$ that appears in Equation 5

Algorithm 1: Iterative computation of the approximate complexity

```
current_explore ← [(M, 0)] ;
future_explore ←[ ] ;
pass ← 0 ;
K(e) ← +∞ ;
while current_explore ≠ [ ] and pass < max_pass do
    for (M_prev, K_prev) ∈ current_explore do
        for π ∈ P do
            for k ∈ {0,1}* do
                K_current ← l(π) + l(k) + K_prev ;
                if K_current > max_complex then
                    break ;
                end
                M' ← f_π,k(M_prev) ;
                if M' = {e} then
                    K(e) ← min(K(e), K_current) ;
                else
                    future_explore.append((M', K_current));
                end
            end
        end
    end
    current_explore ← future_explore ;
    future_explore ← [ ] ;
    pass ← pass + 1;
end
```

$K_{exp}(e)$ evaluates the complexity the user, or the system, would expect the occurrence of event e to have, based on their previous knowledge. In our framework, this prior knowledge consists of the base memory M_0. The expected complexity of the event e can be computed with a simple first-order approximation, i.e. estimating the average complexity of "similar events" over the base memory M_0.

Still, there is a difficulty in defining what should be considered *similar* events. Given that we deal with non comparable events, we may define the notion of similarity by referring once again to *predicates*.For a given event e and a given predicate π_k, we define a π_k-neighborhood of e as the set $N_{\pi,k}(e)$ of all other events satisfying π_k.

$$N_{\pi,k}(e) = \{e' \in M_0, \pi_k(e') \wedge e' \neq e\} \tag{14}$$

Now, when considering, for all possible predicates π_k, the corresponding neighborhoods $N_{\pi,k}(e)$, with the convention that $N_{\pi,k}(e) = \emptyset$ if e does not satisfy π_k, we can compute an average expected complexity for e:

$$K_{exp}(e) = \frac{\sum_{\pi,k} \sum_{e' \in N_{\pi,k}(e)} K_d(e')}{\sum_{\pi,k} |N_{\pi,k}(e)|} \tag{15}$$

This definition is consistent with the intuitive idea that more similar events should weigh more in the computation. Indeed, if e' is very similar to e, it will appear in many neighborhoods, since it satisfies most of the predicates that e satisfies. Therefore, it will be present in more terms in Equation 15, and will weigh more in the final result.

D. From memorability to abduction

Abductive inference builds on the computation of the memorability score. *Knowing* that we want to find a cause c for an observed effect e, we try to find the most remarkable event in past memory that is related to e. While our "memorability" score identifies remarkable past events, it does not take into account their relatedness to e.

The knowledge attached to the occurrence of e can be integrated into the description complexity definition by using conditional complexity $K_d(c|e)$: The information contained in e is considered as given, and therefore "free" in terms of complexity. For instance, when looking for a cause for an anomaly in the living-room, other anomalies occurring in the same living-room will be simpler, as the location "living-room" is already known from the observation of the current anomaly.

Formally, we now consider only predicates where knowledge of the effect event e is appended to all programs k: $\pi_{k::e}(c)$, where :: is the append operation. The set of paths obtained with such predicates is noted P_e^∞. This append operation is free in terms of bit-length in the computation of complexity, since the effect event e is an input of the problem. Therefore, we have $L'(\pi_{k::e}) = L(\pi_k) = L(\pi) + L(k)$. We get a definition for the conditional description complexity:

$$K_d(c|e) = \min_{p \in P_e^\infty} \{L'(p), \quad p(M_0) = c\} \tag{16}$$

$$= \min_{p \in P_e^\infty} \left\{ \sum_{f_{\pi,k::e} \in p} L(\pi) + L(k), \quad p(M_0) = c \right\} \tag{17}$$

This new conditional description complexity translates the additional information provided to the system when answering a user's request. It can then be averaged over similar events to compute the expected conditional description complexity, $K_{exp}(c|e)$. From this, we come to the definition of the conditional memorability, which measures how memorable an event c turns out to be in the context of the occurrence of another event e:

$$M(c|e) = |K_{exp}(c|e) - K_d(c|e)| \tag{18}$$

Conditional memorability encapsulates the idea presented as the motivation of this paper: when confronted with a surprising situation, and in the absence of any other source of information, events that appear more memorable than others, with regards to the target event will be selected as potential causes. As such, our conditional memorability score provides a ranking that can be used for abductive inference.

TODO: Mettre un passage ici pour dire que ça résoud le dilemne de la récence présenté dans l'intro

IV. Implementation Example

A. Setup

We considere here an experimental smart home setup. This choice of configuration is motivated by the challenges posed by smart homes for abductive inference: i) as the number of connected devices increases, more events are recorded, making the detection of memorable events more important; ii) smart homes are prone to experience atypical situations, highly dependent on the context, for which pre-established relations might fail to find good abduction candidates. Our choice was also motivated by the existence of previous work involving smart home simulations capable of quickly generating data from which we could extract events and test our methods.

To carry out the simulation, we built custom modules into the existing iCasa smart home simulator [9]. ICasa offers a simulation of autonomic systems that can handle internal communications, the possible insertion of new components at runtime, or the deletion or modification of existing components. We used a basic scenario consisting of a house with four rooms, a single user, and an outdoor zone. All four rooms are equipped with a temperature controller system that monitors and controls heaters (fig. 2).

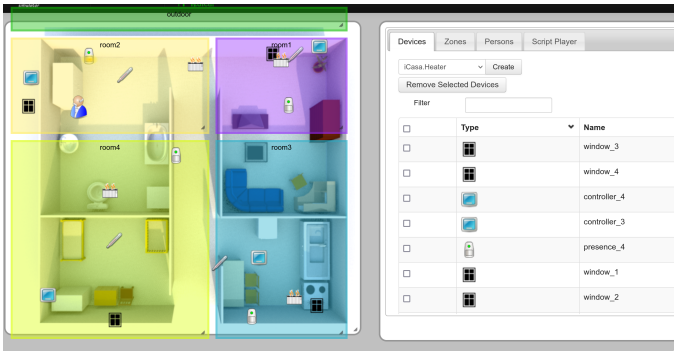

Fig. 2. View of the simulator's web interface provided by iCasa. The four rooms are visible, with their equipment and the user.

Based on this, we implemented a scenario spanning over 420 days, and comprising a daily cycle of outdoor weather (temperature and sunlight) fluctuations, as well as user's movements. All these daily changes create non-noticeable events, serving as a background for our experiments. To produce outstanding events, we randomly generated about twenty events, spanning over the whole duration of the simulation, of different kinds:

- Unusual weather: the outdoor conditions are set to unusually high or low temperatures.
- Heater failures: heater may break down, making them turn off regardless of the command they receive.
- User's absence: the user goes out of the building for an extended period of time.
- Device removal/addition: a device is removed, or another one is added to the system.

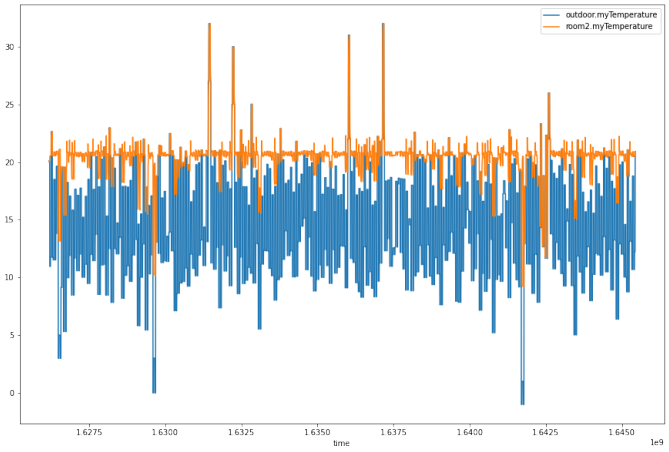

Fig. 3. Time series data from the simulation: outdoor temperature (blue) and controller temperature of a room (orange). On different occasion, the controlled temperature deviates from its setting (21°C). We will evaluate if our system finds these deviations as memorable, and if it can turn them into relevant causal hypotheses for current anomalies.

The values observed from all devices and zones was sampled throughout the simulation run. The resulting data (figure 3) was then used as a basis for our experiments.

B. Implementing the complexity computation

For the implementation of our method, we first needed to identify and characterize events from the time series data generated by the iCasa simulation. Since the selection of events is not the focus point of our present work (see sec. V), we perform event detection merely based on threshold and pre-computed conditions.

This set of events constitutes the basis of the initial memory M_0 used for computations. Then we implemented a small number of predicate concepts:

- $\texttt{label}(\texttt{e}, \texttt{k})$: whether the event e has the label ranked k^{th} in the label list.
- $\texttt{rank}(\texttt{e}, \texttt{r}, \texttt{a})$: whether the event e has the rank r along axis a.
- $\texttt{day}(\texttt{e}, \texttt{k})$: whether the event e occurred k days ago.
- $\texttt{month}(\texttt{e}, \texttt{k})$: whether the event e occurred k months ago.
- $\texttt{location}(\texttt{e}, \texttt{k})$: whether the event e occurred in zone k.

With a straightforward implementation of memory, predicates and filters, we could run alg. 1 worked, though, as expected, it took too long to be usable in realistic scenarios with hundreds or thousands of events to consider. In order to facilitate and speed up computations, we also implemented the following improvements:

- The memory object was augmented with various built-in rankings, allowing for faster operations during future filtering. For instance, since the memory object keeps a mapping from timestamps to events one can perform a quick filtering by date without having to loop over all stored element. This convenient mapping, however, is not directly used to retrieve events by their date of

occurrence, so as to preserve the theoretical model of memory as an unordered set, as presented in section III.

- Each of these predicates holds the property that, in addition to `True` and `False`, they can return another value, `None`, which is theoretically treated as `False` but carries the additional information that this predicate concept will also be false for any other element of the memory for any subsequent program k. This allows to effectively break the innermost loop in alg. 1.
- Some of the filters, for instance the date and rank filters, were hard-written. Events can be selected from these pre-computed mappings over the memory objects rather than by testing a predicate over all memory elements.

C. Results

1) Memorability: The results of the description complexity evaluation, for the described setup, are shown in fig. 4. The entire computation, over 4 iterations (meaning that retrieval paths contained at most 4 filters), took around 30 seconds on a commercial laptop with an i7-8700u CPU.

The blue line emerges from a set of "usual" events, whose complexity varies as the logarithm of the elapsed time since their occurrence. It corresponds to events for which the best retrieval path consists of a time description (e.g. "2 months and 12 days ago"). On the other hand, some events stand out in terms of complexity: some appear simpler, as they can be distinguished by using their rank along some axis ("the hottest day", "the second longest user's absence"), or the rare occurrence of their kind ("the only fault on the heater").

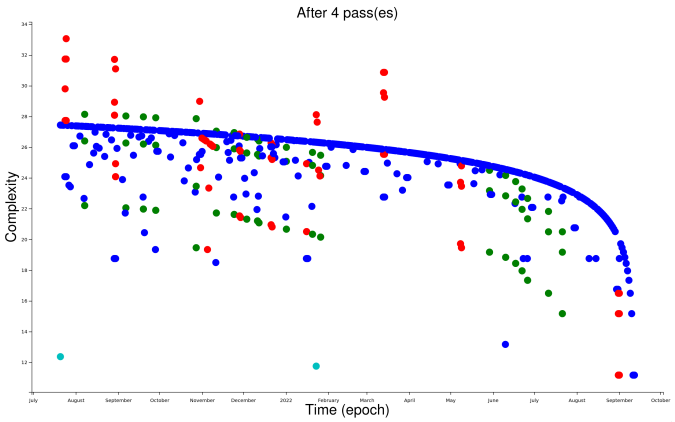

Fig. 4. Computed description complexities of events with retrieval paths of length at most 4. Events of type "day" (blue), "hot" (green), "cold" (red), "device removal"(cyan) are shown.

The "memorability score" is shown in fig. 5. the computation of this measure treats unusually complex or unusually simple events the same way (from the absolute value operation in Equation 4), we observe that some events are memorable due to their context only. For instance, a temperature anomaly occurring simultaneously to many other anomalies is noticeable, as it is costlier than expected to distinguish it from its neighbors.

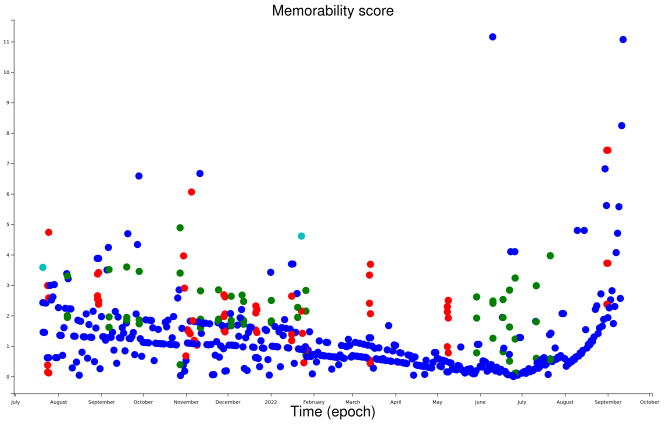

Fig. 5. Memorability score for events in memory

Given that we generated the data used for this experiment, it is possible to flag all perturbation events from the usual daily events and evaluate how a detection based on "memorability" score would perform in distinguishing these events. The result is presented as a ROC curve in fig. 6.

TOOD: Il faudrait commenter la courbe

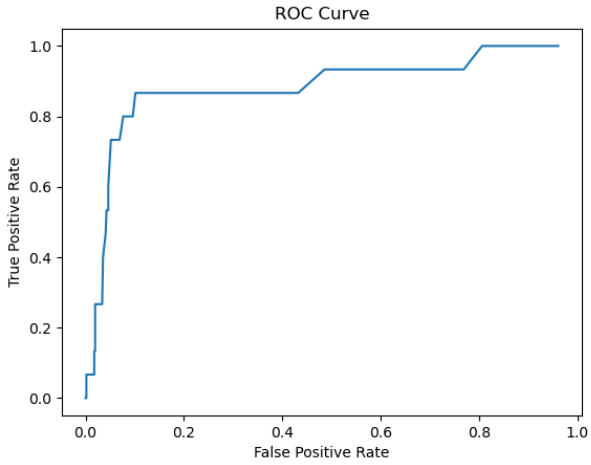

Fig. 6. Experimental ROC curve (True Positive Rate against False Positive Rate) for a classifier based on our memorability score. Measures consider 23 manually flagged events as memorable (events added to the normal background as described in Section IV.)

2) Abduction: As an illustration of the abductive inference made possible our memorability score, we used as a target anomaly the temperature drops visible in the raw time series data from our scenario (fig 3).

TODO: the experiment still needs to be done: some additional data is required here (creating the scenario and finding causes.)

3) Discussion:

V. Related Works

TODO: this section is still WIP: I've added most references and short comments, but I still need to write

the story linking them. Our work is intended to be integrated into larger-scale frameworks to monitor and detect events in complex environment such as smart homes. In these works, the approach to smart homes is often regarded as self-organizing systems [10], [11]. As such, they present capacities of adaptation to new goals, new components, new environment. A commonly used approach is the principle of autonomic system, which minimizes user's intervention for management of the system [11], [12].

We want to inscribe our work among other classical concurrent approaches to abduction. In situations where more data is available, we could for instance rely on correlation or causal inference from known relations [13], [14]. Previous relations between inference and complexity have been studied. In fact, the case of inference was one of the motivations for R. Solomonoff to introduce his universal algorithmic probability [15] as a tool to reach an idealized inference machine, creating the notion of complexity concomitantly to Kolmogorov. Subsequently, notions of complexity re-emerged in causal inference: [16] found, when a causal link exists between two random variables, considering the direct joint probability is simpler, in terms of Kolmogorov complexity, than the inverse direction.

More recently, [17] used Minimum Description Lenght to determine, given a joint probability distribution over (X, Y), whether X causes Y or Y causes X. Their method is based on tree models, and implying that a model respecting the causal relation will be simpler to describe.

[18] PACK algorithm to cluster data using the simplest possible trees: maybe link this to isolation forests?

The topic of finding events from streams of time series data has already been explored in many ways. [19] provides good review of modern approaches and techniques in the field. Some previous work can also be noted for having used AIT techniques to qualify and detect events in time series data. For instance, [20], [21] propose weighted permutation entropy as a proxy for complexity measures in time series data, and use it to find relations between different time series. [22] proposes a MDL approach to find the intrinsic dimensions of time series. All these approaches are interesting, and can be integrated into our canvas as tools to detect events using only complexity. As such, one could acheive a purely complexity-driven process for detecting and qualifying events.

Among various methods, the approach of "Isolation forests" [23], [24] is closely related, by design, to ours. It evaluates the isolation of data points by constructing random binary tree classifiers. On average, outlier points will require less operations to be singled out. Using the average height of leaves in the tree as a metrics, this approach succeeds in identifying outlier points without having to define a "typical" point. This approach can be understood in terms of complexity: each node of binary tree classifier needs a fixed amount of information to be described (which variable and threshold are used). So nodes that place higher need less information to be described. As such, outliers need less information to be singled out. Compared to ours, this method is tailored for data-points living in the same metric space. By using predicates as a proxy for complexity computation, our methods is more general, as it is agnostic regarding the nature of events. However, the introdcution of predicates adds a subjectivity in the determination of memorable points, as we will discuss later.

While all these works advocate in favor of a strong link between complexity and the discovery of causes, not other works extends the notion up to the point we propose in this paper, using complexity only as a tool to express the intuitive notion of memorability, and using it for inference.

VI. Perspectives

The main purpose of the present paper is to show the possible connections between existing definitions of simplicity from cognitive science, Algorithmic Information Theory and a practical use case in cyber-physical systems. As such, many further improvements can be done to pave the way towards a better integration and performance for anomaly detection or abduction.

First, the main limitation of the current approach is the requirement of predefined predicate concepts, from which the different filters are constructed. As an extension, we suggest that in the future, we could explore online generation of such predicate. A possibility would be to analyze discriminating dimensions of incoming data and create predicate as to name these differences, similar to the contrast operations proposed in [25], [26]. For instance, the predicate concept `hot` can be discovered by discriminating a recent hot day along the temperature axis and naming the difference with the prototypical day.

While the execution time is not part of the theoretical view of complexity, it is of prime importance for practical applications, especially when one considers implementation into real-time systems or embedded devices. While the computation we propose appear to be heavy, and possibly heavier as the number of allowed predicates grows, significant time savings can be achieve by trimming the base memory of past events deemed the most "non memorable". For instance, one can only retains the 100 most memorable events from the past. The difficulty with this approach is that such operations should be done in a manner to not interfere with the complexity computations for new elements: by forgetting some past events, even uninteresting ones, one should make sure to keep track of what made the interesting ones, interesting. Investigation of how to do so can pave the way towards practical implementations and dynamic selection of interesting events and help reducing the memory and computation cost of data-driven applications.

VII. Conclusion

References

[1] L. Magnani, *Abduction, reason and science: Processes of discovery and explanation*. Springer Science & Business Media, 2011.

[2] M. Li, P. Vitányi, and others, *An introduction to Kolmogorov complexity and its applications*. Springer, 2008, vol. 3.

[3] J.-L. Dessalles, "Coincidences and the encounter problem: A formal account," *arXiv preprint arXiv:1106.3932*, 2011.

[4] J.-P. Delahaye and H. Zenil, "Numerical evaluation of algorithmic complexity for short strings: A glance into the innermost structure of randomness," *Applied Mathematics and Computation*, vol. 219, no. 1, pp. 63–77, 2012.

[5] A. N. Kolmogorov, "Three approaches to the definition of the concept "quantity of information"," *Problemy peredachi informatsii*, vol. 1, no. 1, pp. 3–11, 1965, publisher: Russian Academy of Sciences, Branch of Informatics, Computer Equipment and ....

[6] P.-A. Murena, M. Al-Ghossein, J.-L. Dessalles, A. Cornuéjols, and others, "Solving Analogies on Words based on Minimal Complexity Transformations," in *International Joint Conference on Artificial Intelligence*. IJCAI, 2020.

[7] N. Chater and P. Vitányi, "Simplicity: A unifying principle in cognitive science?" *Trends in cognitive sciences*, vol. 7, no. 1, pp. 19–22, 2003, publisher: Elsevier.

[8] J.-L. Dessalles, "The Pisa Tower effect." [Online]. Available: https://simplicitytheory.telecom-paris.fr/Pisa.html

[9] P. Lalanda, E. Gerber-Gaillard, and S. Chollet, "Self-Aware Context in Smart Home Pervasive Platforms," in *2017 IEEE International Conference on Autonomic Computing (ICAC)*, 2017, pp. 119–124.

[10] J. Kramer and J. Magee, "A rigorous architectural approach to adaptive software engineering," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 183–188, 2009.

[11] S. Kounev, P. Lewis, K. L. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, and others, "The notion of self-aware computing," in *Self-Aware Computing Systems*. Springer, 2017, pp. 3–16.

[12] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003, publisher: IEEE.

[13] J. Peters, D. Janzing, and B. Schölkopf, *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

[14] K. Fadiga, E. Houzé, A. Diaconescu, and J.-L. Dessalles, "To do or not to do: finding causal relations in smart homes," in *202 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2021, _eprint: 2105.10058.

[15] R. J. Solomonoff, "A formal theory of inductive inference. Part I," *Information and control*, vol. 7, no. 1, pp. 1–22, 1964, publisher: Elsevier.

[16] D. Janzing and B. Schölkopf, "Causal inference using the algorithmic Markov condition," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5168–5194, 2010, publisher: IEEE.

[17] A. Marx and J. Vreeken, "Causal inference on multivariate and mixed-type data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 655–671.

[18] N. Tatti and J. Vreeken, "Finding good itemsets by packing data," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 588–597.

[19] C. C. Aggarwal, *Outlier Analysis*. Springer International Publishing, 2017.

[20] G. E. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series," in *Proceedings of the 2011 SIAM international conference on data mining*. SIAM, 2011, pp. 699–710.

[21] B. Fadlallah, B. Chen, A. Keil, and J. Príncipe, "Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information," *Physical Review E*, vol. 87, no. 2, p. 022911, 2013.

[22] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh, "Discovering the intrinsic cardinality and dimensionality of time series using MDL," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, 2011, pp. 1086–1091.

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[24] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended Isolation Forest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1479–1489, 2021.

[25] J.-L. Dessalles, "From conceptual spaces to predicates," in *Applications of conceptual spaces: The case for geometric knowledge representation*, F. Zenker and P. Gärdenfors, Eds. Dordrecht: Springer, 2015, pp. 17–31. [Online]. Available: http://www.dessalles.fr/papers/Dessalles_14050201.pdf

[26] P. Gärdenfors, *Conceptual spaces: The geometry of thought*. MIT press, 2004.