

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

Étienne Houzé ^{1,2*}, Jean-Louis Dessalles ², Ada Diaconescu² and David Menga ¹

¹ EDF R&D ; {first}.{last}@edf.fr; 7 boulevard Gaspard Monge, 91120 Palaiseau, France

² Télécom Paris; {first}.{last}@telecom-paris.fr; 19 place Marguerite Perey, 91120 Palaiseau, France

* Correspondence: etienne.houze@telecom-paris.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.

Keywords: Kolmogorov Complexity; Algorithmic Information Theory; Simplicity; Abduction; Surprise

1. 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

Citation: Lastname, F.; Lastname, F.; Lastname, F. What should I notice?. *Entropy* **2021**, *1*, 0. <https://doi.org/>

Received:

Accepted:

Published:

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Submitted to *Entropy* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

39 to combine various event types from different devices to compute a unified metrics of
40 “memorability”.

41 To address this issue, we started from the following intuition: while all events,
42 regardless of their characteristics or nature, can be uniquely described using a combi-
43 nation of quantitative or qualitative qualifiers, the most memorable ones are likely to
44 require less words to be described. “Last year’s hottest day” is simpler than “the 182nd
45 day 7 years ago”. How can we quantify this relative simplicity? We could evaluate the
46 complexity of each description, taking into account both the complexity of the concept
47 words (a date of occurrence, a temperature ranking), and of the arguments (1st hottest,
48 182 and 7). This approach, The resulting values define the *description complexity* of events.
49 Our intuition is that memorable events require simpler and less numerous qualifiers to
50 be unambiguously described than unremarkable ones.

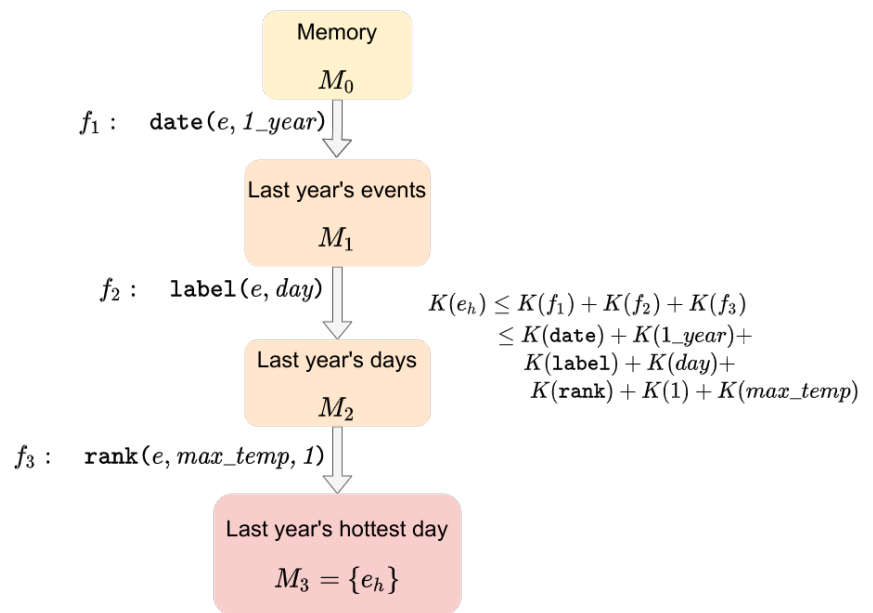


Figure 1. Retrieving an event through successive predicative filters. From the base memory (yellow), successive filters select events satisfying the associated predicate (gray arrows). For example, filter f_1 selects events from last year, i.e. which satisfy the predicate $\text{date}(\text{event}, 1_year)$. 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.

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

We will first briefly introduce some basic notions of Algorithmic Information Theory. Then, in section 2.2, we show how we compute the memorability of past events. We illustrate our approach in (section 3.3) by describing an implementation in a smart home simulation context. In section 4.1 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 4.2

2. Materials and Methods

2.1. Theoretical Background

Kolmogorov complexity formally quantifies the amount of information required for the computation of a finite binary string¹ (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 .

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

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 Equation 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:

$$K_{U'}(s) \leq l(p_U) + K_U(s) \quad (2)$$

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:

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

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 uncomputable[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 $K(e)$.

$$U(e) = K_w(e) - K(e) \quad (4)$$

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

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.

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 at 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 $K_{exp}(e)$:

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

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 well². In the next section, we define computational approximations for the description complexity K_d and the expected complexity K_{exp} of events.

2.2. Computing the memorability of events

2.2.1. 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:

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

where l is the label, t the timestamp and \mathcal{D} a multidimensional 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: $\pi(e, a_1, a_2, \dots, a_n) \mapsto \{0, 1\}$ 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, \dots, a_n . Using this notation, the predicate π becomes a boolean function operating on $\mathbf{E} \times \{0, 1\}^*$:

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

As an example of predicate, consider $\pi = \text{year}$ and k a string encoding the number 1, thus constructing the predicate $\text{year}(e, 1)$, 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*.

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]

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

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

148 For instance, using the same $\pi = \text{year}$ and $k = 1$ as above, we can build the filter
 149 $f_{\pi,k} = \text{last_year}$, which selects all events that occurred last year.

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

$$p = (f_{\pi_1,k_1}, \dots, f_{\pi_n,k_n}) \quad (9)$$

152 By definition $p(M) = f_{\pi_n,k_n}(\dots(f_{\pi_1,k_1}(M)))$. In case the result of the operation
 153 $p(M)$ contains a single element e , we say that the path p *retrieves* the element e from M ,
 154 and write $p(M) = e$. In the example shown in Figure 1, the three filters f_1, f_2, f_3 form a
 155 retrieval path retrieving the event “last year’s hottest day” from the base memory M_0 .

156 2.2.2. Description complexity of events

157 As presented in sec. 2.1, we are interested in computing an approximation of the
 158 description complexity of an event e . From the above definitions, if there is a path p
 159 retrieving e from the base memory M_0 , i.e. $p(M_0) = e$, this path provides a possible
 160 unambiguous description for e . We therefore define the description complexity of e as
 161 the minimum complexity of a path p retrieving e from the base memory M_0 .

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

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

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

$$= L(\pi_1) + L(k_1) + \dots + L(\pi_n) + L(k_n) \quad (12)$$

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

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

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

```

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 β ∈ P do
      for k ∈ {0, 1}* do
        K_current ← l(β) + l(k) + K_prev;
        if K_current > max_complex then
          break;
        end
        M' ← fπ,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

```

Algorithm 1: Iterative computation of the approximate complexity

179 2.2.3. Computing Memorability

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

184 $K_{exp}(e)$ evaluates the complexity the user, or the system, would expect the occur-
 185 rence of event e to have, based on their previous knowledge. In our framework, this
 186 prior knowledge consists of the base memory M_0 . The expected complexity of the event
 187 e can be computed with a simple first-order approximation, i.e. estimating the average
 188 complexity of “similar events” over the base memory M_0 .

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

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

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

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

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

200 2.2.4. From memorability to abduction

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

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

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

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

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

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

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

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

228 This metrics answers the different problems exposed in the introduction: by using
 229 a universal measure for complexity, bits, it allows to compare values from different
 230 dimensions. For instance, it solves the dilemma of recent events: is a big event a long
 231 time ago more memorable than a smaller one only a few minutes ago? The answer from
 232 conditional memorability is that “it depends”. It depends on the complexity cost set for
 233 the predicates describing time and the magnitude of the event. These complexity costs
 234 should reflect the user’s perception of the different dimensions (see Section 4.2).

235 3. Results

236 3.1. Setup

237 We design two different setups to test our approach. Both are inspired from smart
 238 home use cases. This choice of configuration is motivated by the challenges posed by
 239 smart homes for abductive inference: i) as the number of connected devices increases,
 240 more events are recorded, making the detection of memorable events more important;
 241 ii) smart homes are prone to experience atypical situations, highly dependent on the
 242 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.

3.1.1. Scenario 1

In this setup, we aim to reproduce the example mentioned in Section ??: the installation of a brand new smart TV causes unpredictable effects on the light of a room. To play this situation, we created a set of events covering a period of 100 days, corresponding to the past knowledge of the house. Two kinds of events are recorded: “TV event”, corresponding to TV use, and luminosity events, describing the luminosity of the room at a given time. Low lights occur at night, and can occasionally occur during daytime, with a small probability. On the 100th day, a “TV event” is recorded with a different “device” characteristic. Shortly after, the light lowers, which is recorded in the following “light” event.

3.1.2. Scenario 2

We consider an experimental smart home setup, with various sensors, which we simulate over a period of time. 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).

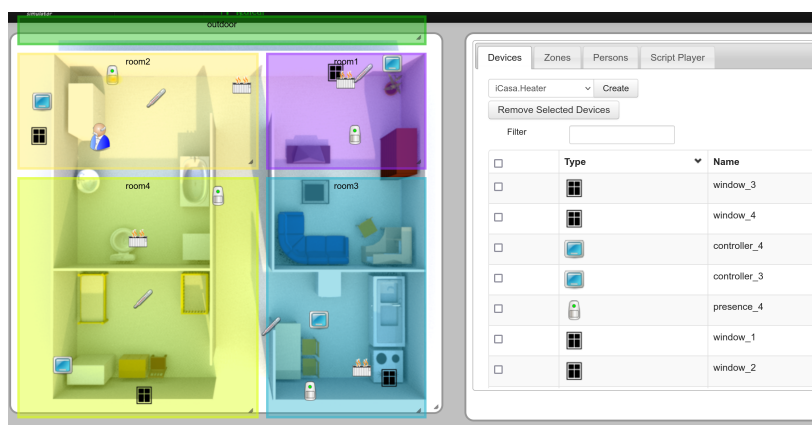


Figure 2. View of the simulator’s web interface provided by iCasa. The four rooms are visible, with their equipment and the user.

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.
- Device removal/addition: a device is removed, or another one is added to the system.

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.

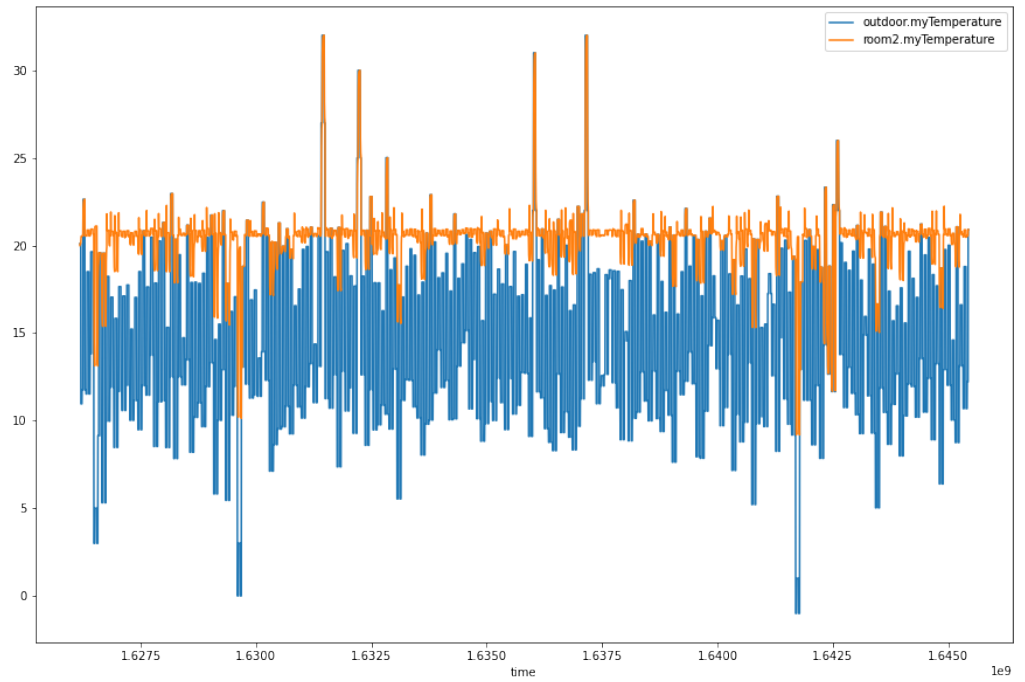


Figure 3. Time series data from the simulation: outdoor temperature (blue) and controller temperature of a room (orange). To be used in our framework, these time series data are processed by a simple threshold-based event detection.

We then process the time series data to identify and characterize events. Since the selection of events is not the focus point of our present work (see sec. 4.1), we perform event detection merely based on threshold and precomputed conditions.

3.2. Implementing the complexity computation

We implemented the computation of both the description complexity and the memorability score into a Python object, called the *SurpriseAbductionModule*. Apart from the base memory of events M_0 , this module contains a set of predefined predicates Π to characterize events. For instance, for scenario 2, the predicates we use are the following:

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

Similar predicates were used for scenario 1, with the notable exception of the addition of a *RandomPick* predicate, which is true if the id of the event matches a random program seed k . Using this predicate and the associated filter in only one of the two illustrative examples shows the importance of allowing random selection in the complexity computation. Furthermore, it could be argued that in some case, random selection of an event is not possible.

The description length $L_{\pi, k}$ of using a predicate is computed as follows: since the set of predicates is finite and known, $L(\pi) = \log 2(|\Pi|)$ are enough to describe the predicate concept π^3 . To encode the argument k of the predicate, we used the prefix-free Elias delta code[10], which requires $L(k) = \log 2(k) + 2 \log 2(\log 2(k) + 1) + 1$ bits. The total cost of describing π_k therefore is

³ This approach gives an equal complexity to all predicate concepts. While this could be argued when using many concepts, as humans do, we deemed better, for our small examples, to consider all concepts as equal.

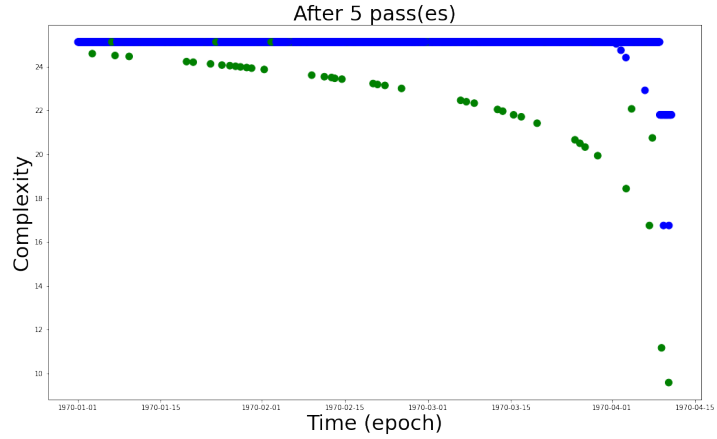


Figure 4. Description complexity for events recorded in the TV scenario. Luminosity events are shown in blue, TV events in green

$$\mathbb{L}(\pi, k) = \log 2(|\Pi|) + \log 2(k) + 2 \log 2(\log 2(k) + 1) + 1 \quad (19)$$

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 timestamp 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 2.2.
- 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 precomputed mappings over the memory objects rather than by testing a predicate over all memory elements.

3.3. Results

3.3.1. The “TV” scenario:

Memorability

The computation of the description complexity measure for the 2400 events recorded in this scenario took around 90 seconds using an i7-8600u laptop. The resulting complexities, in Figure 4, show that TV events appear more simple than luminosity events. In fact, for most luminosity events, the complexity corresponds to the random choice complexity, 25 bits in this example. This pattern comes from the number of luminosity events per day, 24, is too high to consider the retrieval path *day, randomPick* simpler than *randomPick* from the entire memory, except for the most recent days, which appear simpler (on the right-hand side of Figure 4)

Event Id	Description	Relative memorability (bits)
2513	Use of smart TV	16.76
2427	Last use of the old TV	14.81
2411	Second-last use of the old TV	11.21

Table 1: Output of the memorability-based abduction module: top 3 events for the relative memorability metrics.

Abduction

To show the potential application to abductive inference, we use our approach to the first scenario, to see if memorability alone can find the brand new TV to be a reasonable cause for the sudden low light.

The result of our algorithm is given in Table 1. It shows that the system correctly identifies the TV as being the cause. The reason for this choice is that, since the smart TV's device ID is unique among all other events of type "TV", its description complexity is very low.

3.3.2. The "temperature" scenario:

The results of the description complexity evaluation, for the described setup, are shown in Figure 5. 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 general trend appears different from the "TV example" presented in Figure 4: instead of having a capped complexity score, a logarithmic trend appears. This phenomenon is due to the absence, in this example, of the `randomPick` predicate; instead, usual events, with outstanding particularity, can be retrieved by using the `day` predicate and a `label` predicate. As such, it appears that most days are considered usual by our memorability score, creating the main logarithmic sequence of blue dots: there is no better way to retrieve them than simply giving the day of their occurrence. 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 coldest day"), or the rare occurrence of their kind ("the only fault on the heater").

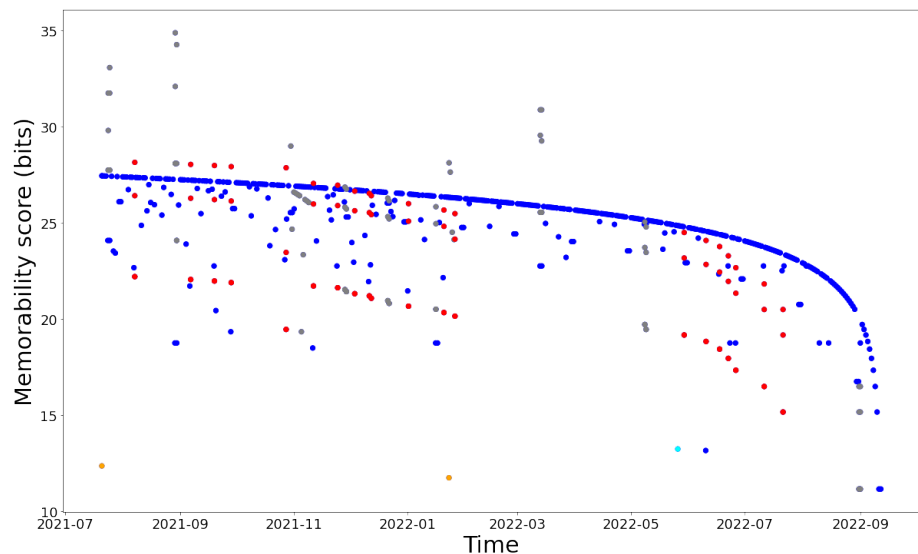


Figure 5. Computed description complexities of events with retrieval paths of length at most 4. Events of type "day" (blue), "hot" (red), "cold" (gray), "device removal" (orange) and "device addition" (cyan) are shown. Events mentioned in the text are specified.

Event iD	Event Type	Retrieval Path
20	day	Label("day"), AxisRank(0, "max_temp")
329	day	Label("day"), AxisRank(0, "min_temp")
183	deviceRemoval	Label("deviceRemoval"), Day(0)
149	cold	Label("cold"), Day(2), Device("thermo_2")

Table 2: Selected events from the “temperature example” with their shortest retrieval path.

The “memorability score” is shown in Figure 6. Similar to what happened with the description complexity score, most events appear with a low memorability: this corresponds mostly to events from the “main sequence” from Figure 5. On the other hand, some events stand out: for instance events 20 and 329, which are respectively the hottest and coldest days recorded, or event 183 which correspond to the rare type *device_removal*. Since our memorability 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, the group to which event 149 belongs appears more complex than expected: the same event occurring simultaneously in all four rooms of the house make each instance harder to discern. Table 2 illustrates this by exhibiting the retrieval paths used for complexity computation for these events.

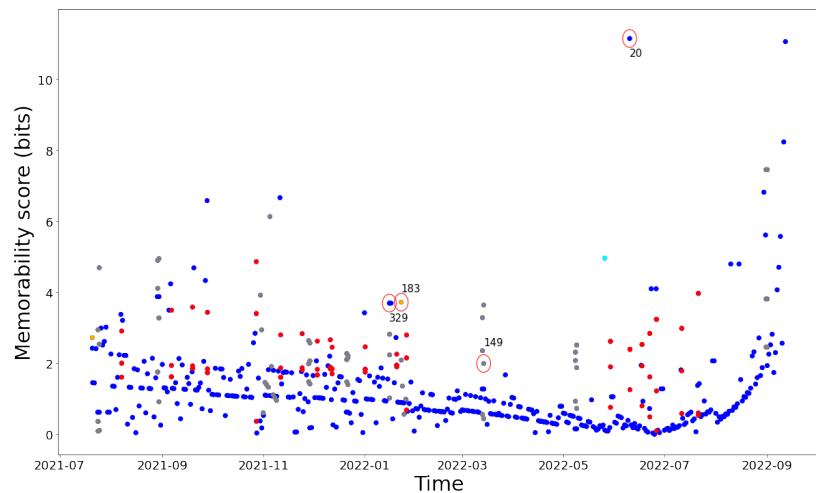


Figure 6. Memorability score for events in memory

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 Figure 7. This shows the memorability score’s performance as a classifying tool for “out-of-the norm” events. In this example, memorability alone correctly identified 18 manually generated events with only 2 false-positives. While the direct application of memorability for classification or event detection is not within the scope of this paper (see Section 4.1 for complexity-based detection), this first result is on par with the motivation of memorability being in accordance with the intuitive notion.

4. Discussion

4.1. Related Works

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 [11,12]. As such, they present capacities of adaptation to new goals, new components, new environment.

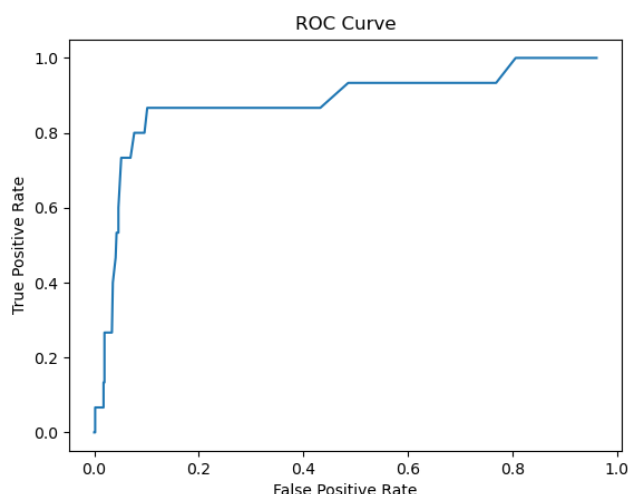


Figure 7. Experimental ROC curve (True Positive Rate against False Positive Rate) for a classifier based on our memorability score. Measures consider 23 manually flagged events as memorable (events added to the normal background as described in Section 3.3.)

384 A commonly used approach is the principle of autonomous system, which minimizes
 385 user's intervention for management of the system [12,13].

386 We want to inscribe our work among other classical concurrent approaches to
 387 abduction. In situations where more data is available, we could for instance rely on
 388 correlation or causal inference from known relations [14,15]. Previous relations between
 389 inference and complexity have been studied. In fact, the case of inference was one of
 390 the motivations for R. Solomonoff to introduce his universal algorithmic probability
 391 [16] as a tool to reach an idealized inference machine, creating the notion of complexity
 392 concomitantly to Kolmogorov. Subsequently, notions of complexity re-emerged in
 393 causal inference: [17] found, when a causal link exists between two random variables,
 394 considering the direct joint probability is simpler, in terms of Kolmogorov complexity,
 395 than the inverse direction.

396 The relation between complexity, compression and causality was used in [18] to
 397 devise the PACK algorithm. It models a dataset by using a family of decision trees where
 398 each tree describes how one variable can be expressed given the others. By choosing
 399 the model minimizing the total description length (i.e. description of the model and
 400 description of the errors), PACK compresses the dataset while finding relations between
 401 variables which can be further analyzed. More recently, [19] used Minimum Description
 402 Length to determine, given a joint probability distribution over (X, Y) , whether X causes
 403 Y or Y causes X . Their method is based on tree models, and implying that a model
 404 respecting the causal relation will be simpler to describe.

405 Another topic where AIT can provide original approaches is event mining in data
 406 streams. [20] provides a good review of modern approaches and techniques in the field.
 407 Some previous work can also be noted for having used AIT techniques to qualify and
 408 detect events in time series data. For instance, [21,22] propose weighted permutation
 409 entropy as a proxy for complexity measures in time series data, and use it to find
 410 relations between different time series. [23] proposes a MDL approach to find the
 411 intrinsic dimensions of time series. All these approaches are interesting, and can be
 412 integrated into our canvas as tools to detect events using only complexity. Using this, one
 413 could achieve a purely complexity-driven process for detecting and qualifying events.

414 The philosophy of our approach can be closely related to the "Isolation forests"
 415 method[24,25]. It evaluates the isolation of data points by constructing random binary
 416 tree classifiers. On average, outlier points will require less operations to be singled out.
 417 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 introduction 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.

4.2. Perspectives

The practical application of the theoretical notions of memorability and relative memorability requires future developments. In this section, we have selected two of them which seem to us, to date, the most challenging.

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 [26, 27]. 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.

5. Conclusions

The introduction of description complexity and the subsequent memorability measure allow to compare events of various types with different characteristics. As such, it provides a formal and generic framework to discriminate peculiar events and outliers.

The computation being based on predicates, the memorability score is entirely dependent on their definitions. This can be seen as a caveat or a feature of our approach: it makes the measure subjective while enabling the consideration of user’s preferences in the definition and choice of predicates. This consideration can be the topic of future research and open up potential applications.

Author Contributions: Conceptualization, É. Houzé and J-L. Dessalles; methodology, software, validation, writing—original draft, É. Houzé; supervision, writing—review and editing J-L. Dessalles, A. Diaconescu and D. Menga. All authors have read and agreed to the submitted version of the manuscript.

Funding: TODO je ne sais pas quoi mettre ici ? Indiquer le financement de la thèse ?

Data Availability Statement: All code and data used for the experiments can be found at https://github.com/EtienneHouze/memorability_code. The iCasa smart home simulator from the Adele research Group, which was used to generate sensor data, can be found at: <http://adeleresearchgroup.github.io/iCasa/snapshot/index.html>

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

References

- Magnani, L. *Abduction, reason and science: Processes of discovery and explanation*; Springer Science & Business Media, 2011.
- Li, M.; Vitányi, P.; others. *An introduction to Kolmogorov complexity and its applications*; Vol. 3, Springer, 2008.
- Dessalles, J.L. Coincidences and the encounter problem: A formal account. *arXiv preprint arXiv:1106.3932* **2011**.
- Delahaye, J.P.; Zenil, H. Numerical evaluation of algorithmic complexity for short strings: A glance into the innermost structure of randomness. *Applied Mathematics and Computation* **2012**, *219*, 63–77.
- Kolmogorov, A.N. Three approaches to the definition of the concept “quantity of information”. *Problemy peredachi informatsii* **1965**, *1*, 3–11. Publisher: Russian Academy of Sciences, Branch of Informatics, Computer Equipment and . . .
- Murena, P.A.; Al-Ghossein, M.; Dessalles, J.L.; Cornuéjols, A.; others. Solving Analogies on Words based on Minimal Complexity Transformations. International Joint Conference on Artificial Intelligence. IJCAI, 2020.
- Chater, N.; Vitányi, P. Simplicity: A unifying principle in cognitive science? *Trends in cognitive sciences* **2003**, *7*, 19–22. Publisher: Elsevier.
- Dessalles, J.L. The Pisa Tower effect.
- Lalanda, P.; Gerber-Gaillard, E.; Chollet, S. Self-Aware Context in Smart Home Pervasive Platforms. 2017 IEEE International Conference on Autonomic Computing (ICAC), 2017, pp. 119–124. doi:10.1109/ICAC.2017.1.
- Elias, P. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory* **1975**, *21*, 194–203. doi:10.1109/TIT.1975.1055349.
- Kramer, J.; Magee, J. A rigorous architectural approach to adaptive software engineering. *Journal of Computer Science and Technology* **2009**, *24*, 183–188.
- Kounev, S.; Lewis, P.; Bellman, K.L.; Bencomo, N.; Camara, J.; Diaconescu, A.; Esterle, L.; Geihs, K.; Giese, H.; Götz, S.; others. The notion of self-aware computing. In *Self-Aware Computing Systems*; Springer, 2017; pp. 3–16.
- Kephart, J.O.; Chess, D.M. The vision of autonomic computing. *Computer* **2003**, *36*, 41–50. Publisher: IEEE.
- Peters, J.; Janzing, D.; Schölkopf, B. *Elements of causal inference: foundations and learning algorithms*; MIT press, 2017.
- Fadiga, K.; Houzé, E.; Diaconescu, A.; Dessalles, J.L. To do or not to do: finding causal relations in smart homes. 202 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2021. _eprint: 2105.10058.
- Solomonoff, R.J. A formal theory of inductive inference. Part I. *Information and control* **1964**, *7*, 1–22. Publisher: Elsevier.
- Janzing, D.; Schölkopf, B. Causal inference using the algorithmic Markov condition. *IEEE Transactions on Information Theory* **2010**, *56*, 5168–5194. Publisher: IEEE.
- Tatti, N.; Vreeken, J. Finding good itemsets by packing data. 2008 Eighth IEEE International Conference on Data Mining. IEEE, 2008, pp. 588–597.
- Marx, A.; Vreeken, J. Causal inference on multivariate and mixed-type data. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2018, pp. 655–671.
- Aggarwal, C.C. *Outlier Analysis*; Springer International Publishing, 2017.
- Batista, G.E.; Wang, X.; Keogh, E.J. A complexity-invariant distance measure for time series. Proceedings of the 2011 SIAM international conference on data mining. SIAM, 2011, pp. 699–710.
- Fadlallah, B.; Chen, B.; Keil, A.; Príncipe, J. Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information. *Physical Review E* **2013**, *87*, 022911.
- Hu, B.; Rakthanmanon, T.; Hao, Y.; Evans, S.; Lonardi, S.; Keogh, E. Discovering the intrinsic cardinality and dimensionality of time series using MDL. 2011 IEEE 11th International Conference on Data Mining. IEEE, 2011, pp. 1086–1091.
- Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 413–422. doi:10.1109/ICDM.2008.17.
- Hariri, S.; Kind, M.C.; Brunner, R.J. Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering* **2021**, *33*, 1479–1489. doi:10.1109/TKDE.2019.2947676.
- Dessalles, J.L. From conceptual spaces to predicates. Applications of conceptual spaces: The case for geometric knowledge representation; Zenker, F.; Gärdenfors, P., Eds.; Springer: Dordrecht, 2015; pp. 17–31. doi:10.1007/978-3-319-15021-5_2.
- Gärdenfors, P. *Conceptual spaces: The geometry of thought*; MIT press, 2004.