On the Notion of Naturalness in Formal Modeling

Eduard Kamburjan[1] and Sandro Rama Fiorini[2]

[1] University of Oslo, Norway
eduard@ifi.uio.no
[2] IBM Research, Brazil
srfiorini@ibm.com

Abstract. We investigate what it means for a formal model to be natural using theories from cognitive science and linguistics. Intuitively, naturalness describes that the formal model fits the domain it is modeling – it is not an intrinsic property of the formal model, but a property that is assigned to it by some human interpreter who is making sense of it. Our main observation is that for each formal model, two sense-making processes are possible: First, the process that interprets the formal model as a symbol in the application domain and assigns it a domain concept. Second, the process that interprets the formal model as a symbol in the engineering domain and assigns it a concept describing an engineering view. Naturalness is described as the similarity of these two mental concepts, i.e., the cognitive complexity to map the domain concept to the engineering concept. We discuss these ideas and formalize then using conceptual spaces, a similarity-based concept representation theory based on cognitive semantics.

1 Introduction

Formal methods is a research field spanning programming languages, logics and other formalisms. At its core, it provides tools for *formal modeling*, the development of a formal representation of a system or a design, as well as tools for its analysis. Despite the highly impressive machinery developed to analyze models, there is little research on the modeling process itself. This is far from being an irrelevant aspect – to-date, there is no way to precisely justify modeling decisions and argue why one modeling language is more suited for one task over another.

Modeling studies are notoriously imprecise in this point, resorting on vague descriptions that core concepts of the modeled domain are "naturally" expressed in the chosen language or that the modeling language has a small "representational distance" (Stehr and Meseguer, 2004; Johnsen et al., 2018) or "is a good match" (Kamburjan et al., 2018) for the domain. The common idea expressed is that the mental concept of the modeler and the concept as formalized in the language are somehow similar. These arguments are not about the mathematical structure – at their core they are arguments about cognition: a formal model is "natural", if the mental process to recognize the structure of the domain in the formal model requires little cognitive effort.

The contribution of this work is a cognitive view on formal modeling to give a framework that allows us to reason about modeling decisions and about cognitive processes during modeling. As our main focus, we make the notion of naturalness more precise. To do so, we fix naturalness as the cognitive complexity of the mapping from the mental concept that arises from interpreting the formal model as an expression in the application domain to the mental concept that arises from interpreting the formal model as an engineering artifact, i.e., based on its functionality. For our framework we draw on three tools from cognitive science and linguistics: (1) semiotics to describe the sense-making processes, (2) conceptual spaces to represent mental concepts and (3) metaphorical mappings to describe the relation of mental concepts.

After introducing the basic ideas behind semiotics in Section 2, we reflect in Section 3 on the classical view on models as abstractions of reality. We then show the inadequacy of the view that the essence of modeling is abstraction to explain why formally equivalent models are perceived with different naturalness. We also distinguish between *non-perceptional* and *perceptional* naturalness – the first is concerned with the similarity of mental concepts, while the later is concerned with more syntactic properties, such as code formatting. The focus here is on *non-perceptional* naturalness.

We aim to provide a way to argue more precisely about modeling, based on our experiences with formal methods – the choice of conceptual spaces is due to their elegant mathematical structure, not a commitment to a model of cognition. We introduce conceptual spaces and their use to describe metaphors in Section 4. The Theory of Conceptual Spaces (Gärdenfors, 2004) is a concept representation framework having conceptual similarity as its main feature. In Section 5 we then use conceptual spaces to make the introduced notions more precise. Additionally, we propose a framework for the mental processes when writing and reading formal models and show where abstraction has its place during the modeling process. Section 6 revisits the examples from previous section and discusses them, and further examples, through the lens of newly introduced framework. Finally, Section 7 concludes.

We do not target models in the broadest possible sense, but instead concentrate on a certain set of scenarios that form the core of use cases for formal models: First, we assume that the formal model has a connection to a domain. Such a connection can be direct, as in formalization of domains with ontologies, or indirect, e.g., in a data model for a database or a programmed application. Second, we distinguish between three roles in the modeling, i.e., the creation of the formal model: (1) the *domain expert* who understands the domain, but little of the formalism used for formal modeling; (2) the *technical expert*, who understands the formalism, but little of the domain; and (3) the *modeler* whose task is understanding both formalism and domain, as well as on communicating with domain expert and technical expert.

A Note on Terminology. Due to the interdisciplinary nature of this work, some of the terms are overloaded, most prominent "model" and "modeling". To avoid misunderstandings we use the term *formal model* for digital or physical artifacts in some formal language and the term *formal modeling* for the cognitive process that produces this artifact. We use the word "concept" for mental models (in contrast to, e.g., Guizzardi (2005), where "model" is used instead). If "model" is used without further specification, "formal model" is meant.

2 Background: A Very Short Primer on Semiotics

Semiotics is the study of *signs*: symbols, their meaning and the processes that connect meaning and symbol. In this section, we give an overview over its main notions, as far as we need them. More precisely, we adopt the triadic model going back to Peirce (1935), with the terminology by Ullmann (1972) and some adaptations to avoid name clashes with computer science notions. For a readable general introduction we refer to Chandler (2017).

For our purposes, a sign is something that is interpreted as signifying something else to somebody. It consists of three components (Fig. 1): (1) a *symbol*, the form the sign takes, which can be, e.g., a word, a sound or an image; (2) a *concept*, the sense made from the sign, in our case a mental concept; and (3) a *thing*, something the sign refers to, which can be, e.g., another sign or some physical domain entity. The sense making process that connects symbol, concept and thing is cognitive and *needs* an interpreter: *"Nothing is a sign unless it is interpreted as a sign"* (Peirce, 1935, 2.172). We stress two details about this model of signs: First, signs are *not* necessarily psychological – while in this work the concept will indeed be a mental concept, this is not true of general signs. Second, signs are triadic – they are *not* the sum of the diadic relations, but arise from the interactions of all three components.

As an example, the word `Tree` is a symbol, the mental representation of trees is a concept and physical trees are a thing. Together, they form a sign. For formal modeling, we can see this triad as follows: the (real or thought-of) system is the thing, the symbol is the formal model of it, and the concept is the mental view of the modeler or reader.

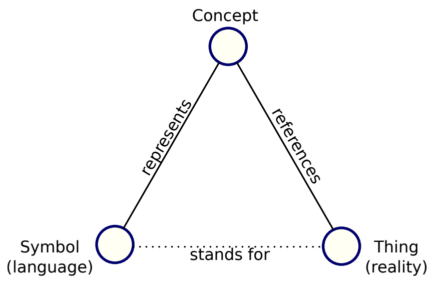


Fig. 1. Adopted semiotic triangle.

We introduce a more precise notion of concept in Sec. 4. For now we use them as an intuitive term for "mental representation used in cognition". Concepts may have properties and have connections to other concepts. For example, the concept "car" has the context of vehicles, the concept "Unit Test" is a concept in Java programming, and the concept "guarded fragment" exists in first-order logic.

Things, i.e., the reality of domain entities, are not a central point in this work. We assume that they exist and that agents can construct conceptual and symbolic descriptions of domain entities. We will rather explore in more detail the representation relation between symbol and concept. Similarly, we do not discuss further that the mental concept is a sign in itself, beyond the observation that the conceptual spaces we introduce in Sec. 4 are symbols on their own.

3 The Inadequacy of Abstraction for Explanation

A formal model is commonly defined along the lines of

> A mathematical representation of a relevant part of a system, design or domain, used for communication and/or a certain analysis.

For example, Peled (2001) defines modeling as *"representing a system in terms of mathematical objects that reflect its observed properties. [...] Modelling usually involves the process of abstraction, i.e., simplifying the description of the system, while preserving only a limited number of the original details."* Definitions of modeling are often mere accessory to the formal method and defined ad-hoc – for a more systematic definition of general models we turn to philosophy, where Stachowiak (1972, p. 131-133)[3] gives the following definition: A model stands for an original (its *Abbildungsmerkmal* or mapping feature), a model is not covering all attributes of the original (its *Verkürzungsmerkmal* or reduction feature, which we identify with abstraction) and a model is standing for its original only for a specific purpose (its *Pragmatisches Merkmal* or pragmatic feature).

In the rest of this work, we understand a *formal model* as a model in the sense of Stachowiak, that is additionally expressed as some mathematical language with formal syntax and an interpretation in terms of other mathematical objects. For example, we consider both programming languages like Java or the λ-calculus, as well as logics as mathematical languages.

Such definitions, which we call the *abstraction-centered*, are emphasizing the the relation of a model to (a possibly perceived or thought-off) reality through abstraction. A model is seen as a clear, partial representation of the world which can be expressed using mathematics and used for a certain intent.

Abstraction-centered definitions are suited to describe what a formal model *is* but are not able to explain why models *are developed* in the way they are in practice. More often than not, models have elements that are unrelated to the things being modeled. For example, consider the Java code in Fig. 2 that models a car with some axles.[4] The class is using a certain serialization framework to

[3] We restrict ourselves to the above setting and do not investigate, e.g., epistemological questions. English translations of the features are taken from Kühne (2006).

[4] We consider Java as being formalized to a sufficient degree to consider it formal.

```
@XmlRootElement(name = "Car")
@XmlType(propOrder={"nrAxels", "name", "pos", "velocity"})
public class Car {
  private int nrAxels;
  private String name;
  private Pair<Integer, Integer> pos;
  private Pair<Integer, Integer> v;

  public Car() {
    this.nrAxels = 0;  this.name = "";
    this.position = null;  this.velocity = null;
  }
  public Car(int nrAxels, String name,
             Pair<Integer, Integer> pos,
             Pair<Integer, Integer> v) {
    setNrAxels(nrAxels);
    ...
  }
  public void drive(Logger log){
    double dist = Math.sqrt(v.component1()*v.component1() +
                            v.component2()*v.component2())
    pos = new Pair<>(pos.component1() + v.component1(),
                     pos.component2() + v.component2());
    log.log(Level.ALL, "Car␣"+name+"␣drove␣"+dist);
  }

  @XmlAttribute
  public int getNrAxels(){ return nrAxels; } // + other getters
  public void setNrAxels(int nrAxels){this.nrAxels = nrAxels;}
  //+ other setters
}
```

Fig. 2. A car with axles as a model in Java.

read and write objects as XML. There are several points that are not explained by abstraction-centered views:

– The method `drive` takes as parameter a logging instance for debugging. Its existence is not related to cars at all.
– The field `nrAxles` has a setter and is initialized with 0. Certainly a vehicle with 0 axles is not a car, and once a car is build the number of axles does not change[5]. Yet, marshaling in Java requires a default constructor and setters for all fields.

[5] We are sure the interested reader can find situations where the number of axles does change in the lifetime of a car. We assume that this class is written for an application that does not consider any of these situations.

There are modeling decisions in this code that we examine in detail in later parts of this article. For now, it suffices to observe that the Java class `Car` is related to the concept of a car through more than abstraction, as it has elements that do not occur in the domain concept. Furthermore, we cannot separate them clearly: the field `nrAxles` should be declared final, given our assumption that the number of axles of a car does not change, but we cannot do that given the marshalling requirements. In that case, a reader of this model might wonder whether that is a feature of the domain or a feature of the language.

This brings us to the core of the problem: a model does not only have a relation to our view on reality, it also has a relation to our view on the formalism. The given code can be read in two ways: The first, which domain experts and car enthusiasts would take, is the one of `Car`-as-cars, an expression of how the modeler thinks about cars. The second, which technical experts would take, is the one of `Car`-as-code, a construct that can be understood by examining how the modeler thinks about Java (and further, technical context). If the technical expert knows nothing about cars, it is as hard for him/her to judge its correct abstraction just as it is hard for the domain expert to detect technical bugs in it. The modeler, given the task to mediate between domain and formalism, is caught in-between: to the modeler, the class is *both* `Car`-as-cars and `Car`-as-code.

This makes the modeling job more difficult: the domain expert is not familiar with XML marshaling – communicating requires the modeler to think of the model *only* as an expression of the relevant parts of the domain concept car. However, the expression is partially formed by the requirements of the formalism and the domain expert must accept that `nrAxles` cannot be final. Communicating with other technical experts requires to think of the model *only* as a concept within the engineering/language domain. This too can be challenging: the modeler may willingly break established design patterns within the formalism so the model is conceptually nearer to the domain.

Let us next make these notions more formal by using the semiotic framework established in the previous section. We introduce three entities: (a) the notion of a symbolic, formal model $\mathfrak{M}$; (b) a conceptual, mental structure $\mathfrak{M}^\delta$ denoting the domain aspects coded in $\mathfrak{M}$; and (c) a conceptual, mental structure $\mathfrak{M}^\alpha$ denoting the engineering, technical aspects coded in $\mathfrak{M}$. For example, considering that $\mathfrak{M} = \texttt{Car}$ then $\mathfrak{M}^\delta$ refers to the conceptualization of notion such as that cars have axles, position and velocity; and $\mathfrak{M}^\alpha$ refers to the conceptualization of a Java class with four attributes, with a `drive` method and so on. The semiotic view is given in Fig. 3: $\mathfrak{M}$ is the symbol for *both* sense-making processes. The first sense-making process (of the domain expert) has the concept $\mathfrak{M}^\delta$ and real cars as things, while the second one (of the engineering expert) has the concept $\mathfrak{M}^\alpha$. The modeler must perform, depending on the current situation perform one of these sense-making processes or possibly switch from one to another. The theoretical question when analyzing this situation is to relate the two sense-making processes.

3.1 The Concept-Centered View on Models

We introduce now the first part of our contribution. Both domain expert and technical expert do not ignore the other's view, but for them one of the views dominates. For the modeler, the domination effect is either not strong, does not exist at all, or shifts depending on the situation. In any case, it is out of the question to ignore one of them. This situation begs the question how the to two sense-making processes of Fig. 3 interact within *one* agent. More precisely, with this view on models, which we call *concept-centered*, the central question is:

> Given a formal model $\mathfrak{M}$, how do $\mathfrak{M}$-as-a-domain-concept ($\mathfrak{M}^\delta$) and $\mathfrak{M}$-as-an-artifact-concept ($\mathfrak{M}^\alpha$) relate to each other?

In this work, we explore what it means for $\mathfrak{M}$ to be *natural*. As a start we say that $\mathfrak{M}$ is natural for a person if $\mathfrak{M}$-as-an-artifact-concept is easy to map on $\mathfrak{M}$-as-a-domain-concept.

We stress that we indeed relate two concepts to each other and stress the difference between $\mathfrak{M}$-as-an-artifact-concept and $\mathfrak{M}$: The former describes the mental representation of $\mathfrak{M}$ by properties from the engineering/formalism domain. The latter is just syntax.

Another aspect that is lost in abstraction-centered views is the choice of the modeling language. This is especially true for programming languages, which are (mostly) all Turing-complete and, thus, equally expressive and have the same pragmatic feature (in the sense of Stachowiak). Abstraction is not able to act as an explanation for the choice of the modeling language for a certain situation, if several languages are able to support the needed analyses. We remind that we understand abstraction as the relation between the model and the modeled thing, not implementation-hiding constructs such as interfaces within parts of the model.

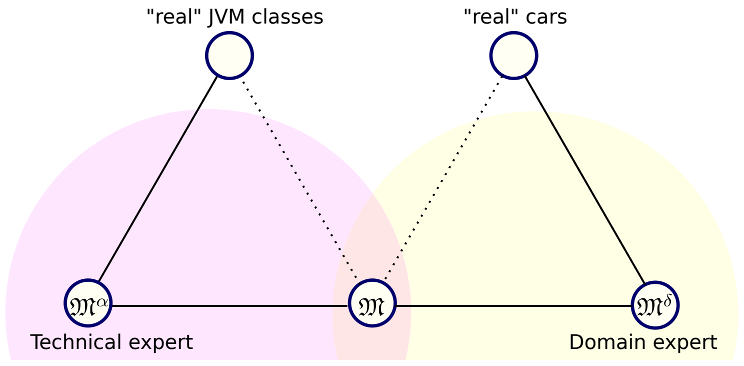


Fig. 3. The concept-centered view on models for two agents.

3.2 Concepts and Syntax

The concept-centered view on models is not antithetical to other views but rather stresses cognitive processes related to modeling. We distinguish between the mental processes concerned with concepts and the mental processes concerned with deriving these mental concepts from perception.

The first class of processes describes the representation of concepts and their processing; e.g., what constitutes a car and what is the relation of the concept car with other concepts. The second class connects these processes with perception; e.g., is the thing that one perceives representing the concept of cars. The distinction is not sharp, but it is useful in our context as it allows us to specifically target syntactic questions. As an example, we give several logic characterizations of the following statement:

A car has 4 wheels.

First, consider the following first-order formula:

$$\begin{aligned}
&\forall x.\ \Big(\mathsf{Car}(x) \rightarrow \exists w_1, w_2, w_3, w_4.\\
&\quad\big(\mathsf{hasPart}(x, w_1) \wedge \mathsf{hasPart}(x, w_2) \wedge \mathsf{hasPart}(x, w_3) \wedge \mathsf{hasPart}(x, w_4)\\
&\quad\ \wedge \mathsf{Wheel}(w_1) \wedge \mathsf{Wheel}(w_2) \wedge \mathsf{Wheel}(w_3) \wedge \mathsf{Wheel}(w_4)\\
&\quad\ \wedge w_1 \neq w_2 \wedge w_1 \neq w_3 \wedge w_1 \neq w_4 \wedge w_2 \neq w_3 \wedge w_2 \neq w_4 \wedge w_3 \neq w_4\big)\Big)
\end{aligned}$$

There are several syntactical effects here that make it easy to grasp what is modeled, before the stage where we can ask whether the model is natural or not. I.e., syntactical effects describe how perceived structures give rise to the model-as-a-concept:

- Conjuncts are grouped together in units that correspond to one statement, e.g., "all parts are wheels" corresponds to the second line,
- the order of these units is almost the same as in the description,
- the literal 4 occurs in both natural language description and formalization,
- indentation gives a clear structure that partitions the unit visually, and
- the variable names clearly related to their intended meaning.

We assume that the predicate and relation symbols are inherently meaningful and that it is not sensible to have the predicate that expresses "x is a car" have any other name. Next, we turn to the units in more detail. The unit

$$w_1 \neq w_2 \wedge w_1 \neq w_3 \wedge w_1 \neq w_4 \wedge w_2 \neq w_3 \wedge w_2 \neq w_4 \wedge w_3 \neq w_4$$

describes that the four variables are different. We imagine that readers familiar with fist-order logic have not read all clauses in detail. Instead the pattern of pairwise inequality is recognized and the unit is perceived as one statement.

Now, consider the following, equivalent first-order formula:

$$\begin{aligned}
&\forall x.\ \Big(\mathsf{Car}(x) \rightarrow \exists \mathit{cake}, \mathfrak{Y}_6, x_2, \mathit{Schiff}. \\
&\quad\big(\ \mathsf{Wheel}(\mathit{Schiff}) \wedge \mathfrak{Y}_6 \neq x_2 \wedge \mathfrak{Y}_6 \neq \mathit{Schiff} \\
&\quad\wedge \mathsf{hasPart}(x, \mathit{Schiff}) \wedge \mathit{cake} \neq \mathfrak{Y}_6 \wedge \mathsf{Wheel}(\mathfrak{Y}_6) \wedge \mathsf{Wheel}(x_2) \\
&\quad\wedge \mathsf{hasPart}(x, x_2) \wedge \mathit{cake} \neq \mathit{ship} \wedge \mathsf{Wheel}(\mathit{cake}) \wedge \mathsf{hasPart}(x, \mathfrak{Y}_6) \\
&\quad\wedge \mathsf{hasPart}(x, \mathit{cake}) \wedge x_2 \neq \mathit{ship} \wedge x_2 \neq \mathit{cake}\big)\Big)
\end{aligned}$$

While one can see it represents the same concept once understood, it is probably harder to derive the mental concept in the first place given how the variables and restrictions are written. We call the distance of concept and formal model *perceptional naturalness*. We will discuss this example in more details when we have formalized perceptional naturalness. Next, we briefly discuss the role of the language in more detail.

Formal models are expressed in some modeling language and the choice of the language plays a role in how a concept is expressed. Thus, the choice of the language has an influence on both the naturalness of the formal model and its perceptional naturalness. We focus on naturalness here, for perceptional naturalness it suffices to note that formal languages such as Whitespace or Malbolge (Olmstead, 1998) and C are all equally expressive languages, yet Whitespace or Malbolge are highly unnatural in any sense of the word.

It is out of scope for this work to discuss computational thinking in detail and to investigate the differences in programming paradigms; e.g., between functional, declarative and imperative programming. Instead, we discuss in more detail how the units introduced in the above example related to the role of language. Returning to the four-wheeled car, we can give an alternative notation for the same formula:

$$\forall x.\ \mathsf{Car}(x) \rightarrow \exists w_1, w_2, w_3, w_4. \bigwedge_{i \in 1..4} \big(\mathsf{Wheel}(w_i) \wedge \mathsf{hasPart}(x, w_i)\big) \wedge \bigwedge_{\substack{i,j \in 1..4 \\ i \neq j}} w_i \neq w_j$$

Is this formula a first-order logic formula? Syntactically it is not, but it is straightforward to expand all the introduced shortcuts and retrieve a "pure" first-order formula[6]. We argue that it is still a first-order logic formula, instead the shortcuts form a *conceptual library* of patterns that are employed anyway. E.g., the grouping into units. Similar conceptual libraries are known in other minimal languages, e.g., the church encoding of the natural numbers in the lambda-calculus. Similarly to programming language libraries, these conceptual libraries are a summary of useful patterns that are repeated in many programs.

[6] Extensions of a simple formalism may be less straightforward than expected, as the study of Quinlan et al. (2019) on the use of BNF grammars in practice shows.

When the modeler starts to express a concept in a certain language, the conceptual (and programming) libraries are included in the expression. I.e., when arguing about the naturalness of a formal model, the libraries must be included in the discussion, as the concept must be adapted to both language and libraries. In this sense, no programming or modeling language is truly minimal, it just gives the user a bigger freedom in the choice of libraries in turn for a higher reliance on these libraries.

3.3 A Complete View on Models

We have so far discussed two views on models: (1) The abstraction-centered view emphasizes the relation of a formal model the thing it stands for. It emphasizes the reduction feature. (2) The concept-centered view emphasizes the dual nature of a formal method and the relation between $\mathfrak{M}$-as-a-domain-concept and $\mathfrak{M}$-as-an-artifact. It emphasizes the mapping feature. For completeness' sake, we mention a third view that emphasizes the pragmatic feature, which we call *purpose-centered*: A model is a mathematical expression made for a certain (business-)purpose. In industrial practice, this view is more relevant than the others: as businesses aim to make money, the availability of trained personal, computational resources, etc. is critical in the choice of what language is chosen and how a model is designed. I.e., one may have a model that is less natural than possible and less abstract than possible, but no employee is able to produce such a model in a reasonable amount of time and the model is good enough for business-purposes.

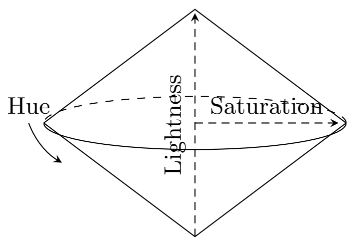
4 Conceptual Spaces

To examine naturalness, we must be able to analyze the relationship between $\mathfrak{M}^\delta$ and $\mathfrak{M}^\alpha$; In search for tools to examine and describe these mental concepts we turn to cognitive science. This research field is, among others and briefly summarized, concerned with explaining and constructing cognitive activity. Cognitive activity uses some information to reach some goals. While the nature of how this information and these goals are represented mentally is an elusive mystery, there are numerous theories of *modeling* representations. The following section, and indeed this whole work, does, thus, *not* claim that the used notion of concepts is “real”, in the sense that the cognitive activity uses conceptual spaces for representation, they are a model *for* the representations.

The theory used here is the Theory of Conceptual Spaces of Gärdenfors (2004, 2014), which is based on geometric structures and motivated existing cognitive phenomena, such as similarity[7]. In short, conceptual spaces (CS) are metric spaces where concepts are represented as regions, objects as points and dimensions are ways in which these can be compared. Similar concepts are grouped

[7] This is in line with a tradition to describe concepts/categories not by common features, but by distance between instances, following Wittgenstein's family resemblance (Wittgenstein, 1953) and Rosch' prototype theory (Rosch and Mervis, 1975).

closer together in a conceptual space. For example, the concept Apple could be represented as a region in a CS where the dimensions are shape, color, and weight. In this space, the region for Oranges would be closer to Apple than Pineapple, for example. The relevance of Conceptual Spaces to our discussion resides in its proposal as a framework for cognitive semantics. In cognitive semantics, the meaning of linguistic expressions is given by *mental* entities, which are grounded in reality through perception. Conceptual spaces provides a structure for such mental entities. That fits our view in which (symbolic) models, such as Java classes, are interpreted as mental models of domain or technical entities.



A considerable part of what makes CS a powerful representation framework is the way in which the dimensions of the metric space are structured. Certain dimensions, particularly perceptual ones, always appear together, forming *quality domains*. For example, hue, saturation, and luminosity (HSL) are integral[8] to each other, forming a color domain (to the right). In addition to that, the Theory specifies that *natural properties* are convex regions in quality domains. The notion of natural property is loosely defined as those natural for the purpose of usual problem-solving tasks. So, for example, the property Red in the HSL space should be convex. Indeed, studies with color perception in different cultures showed that regions for basic colors in HSL space are indeed approximate convex regions (Sivik and Taft, 1994). Studies with other perceptual domains paint a similar picture (Gärdenfors, 2004).

Complex concepts, such as Apple, can be more precisely defined as collections of regions in quality domains. Those include perceptual domains, but can also include non-perceptual domains, such as price and shelf life.

The Theory is not intended to be complete. One important aspect for our discussion and that is not well established is related to conceptual spaces uniqueness: how many conceptual spaces there are in an agent's mind? Some works assume a single a conceptual space with a high number of dimensions in which all concepts are represented (e.g., Aisbett and Gibbon, 2001). Some works assume smaller conceptual spaces, sometimes one for each concept (e.g., Fiorini and Abel, 2013). Furthermore, distinct agents are normally assumed to have distinct conceptual spaces, which can be aligned by mapping conceptual spaces to symbolic structures and then by symbolic communication (Warglien and Gärdenfors, 2013). In this paper, we assume each agent (e.g. modeler or expert) has a collection of subjective conceptual spaces focused in specific topics, which may or may not be result from the projection of a universal subjective conceptual space. We also assume that structures of these conceptual spaces are mapped to language structures, which can be communicated symbolically through written artifacts. Furthermore, we assume that the decoding of these artifacts induce a conceptual space.

[8] I.e., it is not possible to assign a value to an object in one dimension without assigning one in the others.

Other cognitive phenomena can be explained in terms of operations in conceptual spaces. For example, taxonomic reasoning can be defined in terms of region embeddings/projections. Contextual effects can be explained in terms of dimension weighting (see Gärdenfors (2004) for more details).

Concept Composition. From the tools developed for Conceptual Spaces, we require those related to concept composition and introduce these next.

Only in the most simple case can we model the composition of two concepts as their product space, i.e., intersection of the regions in their properties, because only the most simple composition shares properties. This is, for example, the case for "red car", which is composed from "red things" and "car". Such compositions are described by intersecting the region for "red" in the color property of "car" and leaving the rest of the concept of "car" unchanged.

A more common case is that while intersection can be applied to some regions, other regions are *incompatible*. The classical example here is "stone lion". The material property of "lion" has an empty intersection with "stone", so instead the property is *replaced*. However, some of the properties of "lion", namely all which are concerned with living things, are not compatible with things made of stone and are consequently removed from the composed concept. Indeed, the only remaining property of "lion" is its shape.

A similar situation arises in natural language with metaphors, where the composition of concepts cannot be described by removal, addition and interactions of properties, as the concepts share no properties (Lakoff and Johnson, 1980). Metaphors are *"mappings across conceptual domains"* (Lakoff, 1993) that preserve some cognitive structure not directly accessible on a lexical level, instead they *"preserve the cognitive topology (...) of the source domain, in a way consistent with the inherent structure of the target domain."* (Lakoff, 1993).

The similarities between metaphors and models run deeper than their shared property of mapping across domains. Both metaphors and models are, in the words of Steen (2011), *"not a matter of language but of thought"*. A model is a model because it is thought of as such; there is nothing in a Java program or a first-order formula that gives it its mapping feature without a mind to perform the mapping. Similarly, the adequacy of a model is a property that is inherently non-lexical and cannot be judged without a cognitive approach.

To handle such situations in the Conceptual Spaces framework, (Gärdenfors, 2014, Ch. 13) describes two mechanisms: for shared domains, the concept is *projected* on the shared dimensions (in the example above, the lion is projected on its "form" dimension). This is not abstraction, which is removing properties based on the context of the formal modeling. For non-shared domain, a *metaphorical mapping* is used: a homeomorphism between the regions of the two concepts, i.e., an isomorphism preserving structural/topological properties. Through the homeomorphism, structures from one domain can be applied in the other one, which may not posses such structures.

5 Naturalness in Conceptual Spaces

We have so far established a semiotic view on models, argued that naturalness must be explained as a relation between $\mathfrak{M}^\alpha$ and $\mathfrak{M}^\delta$, and introduced Conceptual Spaces and metaphors as a tool to describe such relations between concepts.

Now we can revisit the notions discussed in Sec. 3. To do so, we discuss mental processes that relate artifacts and mental representations, which differ between *enmodeling* as a mental process to generate a mental concept for a given context (which, most likely, is more simple or suited) and *encoding*, for the process that encodes this model in a formal model or symbol.

5.1 Redefining $\mathfrak{M}^\delta$ and $\mathfrak{M}^\alpha$ and other mental models

As we stated earlier, both $\mathfrak{M}^\delta$ and $\mathfrak{M}^\alpha$ are mental representations that base the production/understanding of artifacts $\mathfrak{M}$. Taking Conceptual Spaces as our framework for mental representation, we must then define their nature in terms of conceptual constructs.

We start by introducing $\mathfrak{C}^\delta$ as a region in a conceptual spaces denoting the full conceptualization of a domain concept. It spans properties in perceptual and non-perceptual quality domains representing the overall experience a person might have with exemplars of such concept. For example, for a car expert, $\mathfrak{C}^\delta$ captures aspects related to specialist and common-sense knowledge about cars.

In contrast, $\mathfrak{M}^\delta$ is a region in a conceptual space constructed by domains and subproperties derived from $\mathfrak{C}^\delta$ that are relevant to the task at hand. In our Java car example, $\mathfrak{M}^\delta$ include domains regions related to axles, position and velocity.

Similarly, we introduce $\mathfrak{C}^\alpha$ a region in conceptual spaces denoting one's general knowledge about the constructs and structures in the target formal model. In our Java example, $\mathfrak{C}^\alpha$ equates to the general notion one has of Java classes. Points in this conceptual space denote individual possible Java objects. Its domain structure is more elusive, given its abstract nature. Examples of quality domains include memory position, hash encoding and use of logging.

Finally, $\mathfrak{M}^\alpha$ represents the formalism and task-dependent mental model of the $\mathfrak{M}$. While $\mathfrak{M}^\delta$ might be a property region on a domain denoting the range of possible number of axles a car might have, $\mathfrak{M}^\alpha$ would have a counterpart region on a integer domain with no region defined.

In the following, we use $\mathfrak{C}^\mathrm{x}/\mathfrak{M}^\mathrm{x}$ when δ and α concepts are interchangeable.

5.2 The Place of Abstraction

We next discuss the relation of $\mathfrak{M}^\mathrm{x}$ and $\mathfrak{C}^\mathrm{x}$ and the process involved with their construction. Modeling is a mental process that starts with a concept (i.e., a region in a conceptual space) and ends with an (physical) artifact expressing this concept in a certain context and a certain (formal) language. We hypothesize that it consists of two main steps: *enmodeling* and *encoding*. In reverse, the mental process that starts with an artifact and ends with a concept consists of *decoding* and *demodeling*. Fig. 4 depicts these processes for $\mathfrak{M}^\mathrm{x}$ and $\mathfrak{C}^\mathrm{x}$.

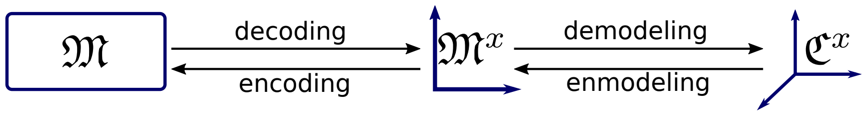


Fig. 4. The mental processes and representations to produce and understand models. The two right representations are cognitive spaces, denoted by blue axes.

Enmodeling. Enmodeling takes $\mathfrak{C}^{\mathrm{x}}$ and adapts it to the target context and target language. Adaptation to the context is mainly abstraction: the removal and rescaling of dimensions and properties that are not relevant for the context. Adaption to the language is a more elusive process: we conjecture it to be similar to the effects known in linguistics that language influences the way concepts are formed and expressed in natural language. The result of enmodeling is again a concept, a region in a conceptual space. Due to the adaption to the language, it is *not* a subconcept of the one we started with.

Encoding. Encoding starts with the adapted concept and ends with the artifact. It is the generation of artifacts from mental concepts *after* adaption of the concepts. These two processes are not independent: obviously, enmodeling is influenced by the target language and is guided by the expression of concepts in this language. Their relation is also not necessarily sequential, but we conjecture that enmodeling starts before encoding, and that encoding ends after enmodeling. For our discussion, it suffices to regard them as separate and ordered.

Decoding. Decoding starts with the artifact and ends with a adapted concept that is specific to its context (i.e., the application in question where the artifact is used). It is the opposite of encoding and the resulting concept still contains traces of the artifact, as it is done in a certain language and context (which the ending concept is adapted to). Decoding contains numerous mechanisms, for example it is the part that is concerned with *perception*. It also may involve higher cognition mechanisms, like memory.

Demodeling. Demodeling is the opposite process of enmodeling, it starts with a concept that contains traces of the artifact language and ends with a full, not context-specific context. It relates $\mathfrak{M}^\delta$ to a $\mathfrak{C}^\delta$ and has, as one of its main parts, the task to recognize the modeled concept in the artifact. For example, it is during demodeling, when the car expert recognizes the Java class as a *specific* concept from the car domain. The moment when the car enthusiast recognizes the Java class as something from the car domain is during decoding. The main difference of enmodeling and demodeling is their direction w.r.t. complexity of the concept: enmodeling reduces complexity (e.g., by removing a dimension), while demodeling increases it.

Demodeling and enmodeling are not monolithic processes and contain subprocesses which may take the opposite direction w.r.t. complexity as the overall process. The exact subprocesses are, however, not of importance for the phenomena we aim to describe here. The important detail is that (de/en)modeling and (de/en)coding can be distinguished: (de/en)modeling is an internal transformation of mental concepts, while (de/en)coding is their relation with the artifact.

5.3 Formal Models as Metaphors

The end of the decode-demodeling process is a concept independent of the artifact, but in our setting with domain view and engineering view, there are two process with *one* beginning, namely the artifact, and *two* ends.

This means that each involved person, engineer and domain expert, have their own decoding and demodeling process when examining a single formal modeling artifact. Not only are the processes different: the concepts are different as well. At the end of the engineer's demodeling, the concepts describe the artifact in purely technical term. For example, it describes the class in terms of properties of Java classes (e.g., final or not, number of fields) and is, in the extreme case where the engineer has no knowledge about the modeled domain, free of any domain dimensions.

If the artifact is examined by an engineer and a domain expert, the artifact essentially becomes a message. Here, we are however interested in our modeler, for whom the artifact is a concept in both the domain context and the technical context. For enmodeling, the modeler also starts with $\mathfrak{C}^\delta$. It is not possible to start with $\mathfrak{C}^\alpha$, as the main task is to model a domain situation, not to produce (some) working code. For encoding, the modeler needs to operate on $\mathfrak{M}^\alpha$, as for this task the technical knowledge is dominant. Thus, $\mathfrak{M}^\delta$ is an *intermediate* concept, constructed during enmodeling.

We refine our view on enmodeling into two steps: (1) *abstraction*, a process that generates $\mathfrak{M}^\delta$ from $\mathfrak{C}^\delta$ by adapting it to the application scenario and (2) *adaptation*, a process that reformulates $\mathfrak{M}^\delta$ by relating it to the chosen language and technical framework. Similarly, when reading a model, the modeler decodes the artifact into $\mathfrak{M}^\alpha$, then disperses the technical framework to arrive at the abstracted domain concept $\mathfrak{M}^\delta$ and finally relates it to the final concept $\mathfrak{C}^\delta$.

Note that $\mathfrak{C}^\alpha$ is not constructed in this process. However, the modeler is able to suppress the domain side and construct $\mathfrak{C}^\alpha$ directly from $\mathfrak{M}^\alpha$, i.e., act as a technical expert.

As discussed, these extreme views of the technical expert and the domain expert are unlikely to occur. Every domain expert has some basic linguistic knowledge and *must* be able to construct some $\mathfrak{M}^\alpha$. However, due to the lack of technical knowledge, $\mathfrak{M}^\alpha$ is rudimentary – it is comparatively hard to disperse the language specifics of the model to reveal the underlying domain structures.

Metaphors. We see that formal modeling requires to compare the structure of different concepts. Following up on on Lakoff's observation that metaphors are mappings across conceptual domains that preserve cognitive topology, we also see that formal models are merely metaphors themselves: The structure of $\mathfrak{M}^\delta$ must be preserved in $\mathfrak{M}^\alpha$. To define naturalness we can, thus, use the mechanisms

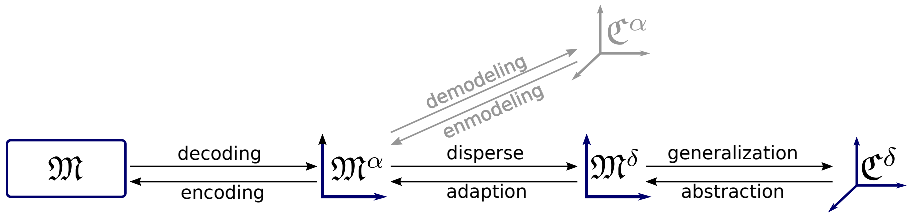


Fig. 5. The mental processes and representations for the formal method expert.

already discussed for concept composition and metaphors through conceptual spaces.

It is interesting to note that when we see models as metaphors, we use the structure of the formal model to explain effects in the domain. This is the opposite direction from metaphors in everyday use in computer science, which use the structure of some "domain" to explain the formal model. For example, the notion of a *stack* (Colburn and Shute, 2008) uses the structure of the domain (being able to add on top) to illustrate the computational concept.

Signs. We have now two mental concepts for each sense-making process: the "raw" concept and the adapted concept. The semiotic triad we use to introduce semiosis, however, has no place for the adapted concept. Our solution is that adapted concepts play a role in two sense-making processes, as illustrated in Fig. 6: the adapted concepts. $\mathfrak{M}^\alpha$ and $\mathfrak{M}^\delta$ are a *concept* for the first sense-making process (the one for en-/decoding) and a *symbol* for the second sense-making process (the one for en-/demodeling). Such a sequence of two sense-making processes which share the same thing and where one concept is the symbol of the other, can be seen as an instance of "successive interpretants"(successive concepts) in the Peircean theory of signs.

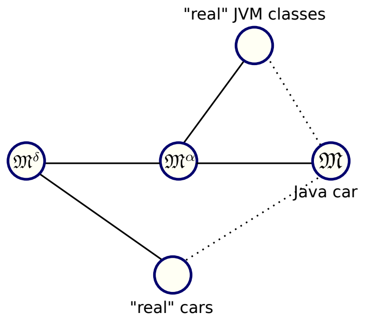


Fig. 6. The semiotic relationships for formal models.

5.4 Naturalness and Perceptional Naturalness

The previous section established a framework for the cognitive processes for formal modeling. Now we use the ideas from cognitive linguistics on concept composition to define of notions of naturalness and perceptional naturalness. Intuitively, a model is natural if it is a good metaphor: there is a metaphorical mapping from $\mathfrak{M}^\alpha$ to $\mathfrak{M}^\delta$ that requires little cognitive effort to map structures from the artifact-view to the domain-view. We now make our notion of naturalness more precise:

> Let $\mathfrak{M}$ and $\mathfrak{N}$ be two formal models for the same aspect of a domain, i.e., the same $\mathfrak{M}^\delta = \mathfrak{N}^\delta$. Let μ be a metaphorical mapping from $\mathfrak{M}^\alpha$ to $\mathfrak{M}^\delta$ and ν the corresponding metaphorical mapping from $\mathfrak{N}^\alpha$ to $\mathfrak{N}^\delta$. We say that $\mathfrak{M}$ is more natural than $\mathfrak{N}$ if μ has a lower cognitive complexity than ν.

In short: a model is more natural than another if it is easier to recover the domain conceptual structure from the artifact conceptual structure. Cognitive complexity denotes the effort needed to perform the metaphorical mapping, which in our setting we interpret as the computational complexity of the metaphorical mapping, if we see the metaphorical mapping as a function between two metric spaces. Using the computational complexity in cognition has a certain appeal when comparing the mind with computers (van Rooij, 2008), but here we do not use it for general assumption about cognition, but to measure the complexity of a certain cognitive task. For a more detailed discussion on computational complexity effects in cognition we refer to Isaac et al. (2014).

In this setting, the complexity of the metaphorical mapping is harder to grasp, as several components are moving: both conceptual spaces are influenced by mental changes, i.e., they change their shape through learning. For example, the space for the language gains more dimensions as the domain expert gains more experience with it. Furthermore, operations within the mapping become computationally cheaper if they are performed more often – analogously to results in natural languages, where less frequent language fragments have higher complexity (in terms of the logic needed to formalize it) (Thorne, 2012). Thus, the computational model in terms of needed operations may also change over time. Consider the example of the first-order logic formula formalizing a car with four wheels. The unit describing that the four wheels are different requires conscious reading of all conjuncts for the novice, i.e., a *linear* complexity in the length of the formula, but after more exposure to the usual patterns in logical modeling, this is reduced into one reasoning step, i.e., *constant* complexity.

If we fix a threshold for low cognitive complexity, then we can give a definition of naturalness that does not require a second model to compare with.

> Let $\mathfrak{M}$ be a formal model. Let μ be the metaphorical mapping from $\mathfrak{M}^\alpha$ to $\mathfrak{M}^\delta$. We say that $\mathfrak{M}$ is natural if μ has a low cognitive complexity.

If we accept the P-cognition thesis that "cognitive capacities are limited to those functions that can be computed in polynomial time" (van Rooij, 2008), than, in our eyes, a sensible assumption would be that μ is natural if it is even less complex than polynomial. One obvious candidate would be linearity, but we leave this question open.

Let us return to the example in Fig. 2, in particular the number of axles. Let us assume that in $\mathfrak{M}^\alpha$, the field of a Java class, is represented as a property with the dimensions "type" = integers, "modifier"=private and "name"=nrAxles. Note that nrAxles here is purely symbolic and not connected to the concept of axles at all. In $\mathfrak{M}^\delta$ the number of axles is just a single dimension over an interval in the natural numbers.

The mapping μ is of low cognitive complexity: exactly one property is mapped onto directly one dimension and both the name and type of the field are directly related to the domain dimension. In terms of concept composition, one can simply perform a *property replacement*. The additional dimension of the modifiers in the engineering domain can just be removed in the mapping – it must not be disentangled from the other dimensions.

Now consider a Java class where the number of axles is modeled as following:

```
private int wheels;
private int wheelsPerAxis;
...
public int getAxles() { return wheels/wheelsPerAxis; }
```

This is less natural: $\mathfrak{M}^\alpha$ now has three different properties and $\mathfrak{M}^\delta$ must include the notion of wheels, i.e., be more precise in its representation of axles. The mapping is more complex: three properties are mapped onto one property. The property of the method is furthermore more complex and involves arithmetic.

Next, let us examine the use of `int` as a type for the number of axles. It is rather unnatural, because it allows to create `Car` instances that cannot be mapped to points in the conceptual space $\mathfrak{M}^\delta$ for car; e.g., those with a negative number of axles. This is obviously not a metaphor: the structure provided by the formal model does not carry over the application domain. It is a consequence of a, possibly conscious, modeling decision to use integers, as these are easily available in the language, while, for example, ranges are not (in Java). This is an example of how none of the processes is performed in isolation – the choice of the target language already influences the enmodeling process. The use of integers is still relatively natural, as (a) integers are often used to overapproximate ranges and (b) one dimension is mapped onto one other.

Finally, we discuss the serialization framework. It is completely foreign to the domain of cars, but some parts are more unnatural than others. The annotations, such as @XmlAttribute are unnatural, but they are easy to ignore – the cognitive mapping just removes the dimensions related to annotations. It is also unnatural that nrAxels can be changed. But while it is also effectively removed in the cognitive mapping, this requires more cognitive complexity compared to the removal of annotations, because it models possible behavior that cannot be *fully* ignored. Indeed, it contradicts the domain – and the cognitive mapping must thus involve more domain reasoning why this contradicting behavior can be ignored. We expect that over time, i.e., after working with the model for some time, memory and association effects will make it more natural as the reason becomes part of the memory.

Similarly, the *cake* variable in our FOL example increases complexity, as it breaks the context: it implies that the context contains notions of baking, which activates the wrong memories and makes it harder to understand what symbols carry domain information and which do not. More generally, syntax highlighting is a technique to increase perceptional naturalness by reducing the cognitive burden required to build the concept $\mathfrak{M}^{\alpha}$.

Perceptional Naturalness. While we support the idea that naturalness is mainly associated with dispersion and adaption mappings, there are formal language features that are more perceptual but that also influence how easily a formal artifact can be understood. Consider the block structure in the first-order example in Section 3 or the use of syntax highlighting. These are features of formal models — some of them *ad hoc* — that helps decoding.

These effects are associated to what we call *perceptional naturalness.* In our view, an artifact is perceptually natural if it has *low decoding complexity.* Since decoding is outside the scope of our Conceptual Space-based framework (decoding is not purely conceptual), we do not investigate decoding complexity further.

Note that the mutability of `nrAxles` above does not fall under perceptional naturalness: the mutability of the field is unnatural, not the absence of a **final** modifier and the presence of the setter. In general we consider most iconicity effects to fall under perceptional resemblance, but iconicity plays little role for the formal languages that we consider here. For example, the only iconicity in the car formula is the occurrence of four different variables for the four wheels.

Further Details. Naturalness is defined in terms of understanding the model, i.e., how easy it is to disperse the language structure, but still related to adaption, i.e., how easy it is to model something: natural pairs of concepts are easy to compose (by replacing the domain properties by engineering properties) and easy to decompose. It is able to explain why something is natural to *express* is a certain way, as well as able to explain why something is natural to *understand.*

Naturalness is more important to the modeler than to the domain expert or the technical expert: the technical expert is not interested in $\mathfrak{M}^{\delta}$, except when it relies on his common sense for explanations[9]. For the domain expert, the domain view is dominant, so the domain expert decodes into $\mathfrak{M}^{\delta}$ *almost directly*, as the domain expert has too few dimensions and properties to build a sensible concept $\mathfrak{M}^{\alpha}$. For the domain expert, there is no sharp difference between dispersal and generalization – naturalness and perceptional naturalness merge.

6 Discussion

Consequences for Interdisciplinary Modeling Studies We once again stress that naturalness is a purely mental notion and, as such, different for every person: it

[9] For example, we can assume any programmer to have some knowledge about cars.

is not possible to reason about the naturalness of an artifact per se, as without an interpreter no sense-making processes occur.

However, we can reason about naturalness is a restricted context beyond a specific person: given a certain domain and application, we can assume that the domain concepts have similar structures for different people working in a field, due to common education and experiences. Thus, if one such person perceives a model as natural, it is likely that this generalizes within the target group.

We can, thus, also make assessments of naturalness of formal modeling *languages*: a language is more natural than another, if there is a more natural model in it. We can approximate this by trying to map the core concepts of the domain on constructs within the existing language. For example, consider a mail service to send letters. It is more natural to model such a service using the actor concurrency model than, let's say, in a shared memory model with semaphores, because the basic language feature of *asynchronous messages* shares structure with *sending a letter*, because both may be reordered and require, in general, no waiting for a response. In contrast, to model the same property requires a more complex formulation when using semaphores.

This is, in essence, the underlying assumptions why domain-specific languages work in practice: if the vocabulary and constructs are fixed and the target group shares education and experiences, then they find it natural to express themselves in it, i.e., to write natural formal models, if the language has primitives for common relations and actions.

It follows from the above that the modeler is *not* able to judge the naturalness of the formal model until the modeler is trained enough to align the domain concepts with the one of established domain experts. This confirms our experiences in the common setting where the modeler starts as a technical expert and acquires domain knowledge until the modeler can take the role of the mediating formal method expert: The first iterations of a formal model are mainly useful to find out where the preliminary intuition of the formal method expert is still wrong. Yet, we found early prototypical models of critical importance to establish a successful interdisciplinary collaboration: these models train both technical expert (i.e., the modeler to-be) and the domain expert to use models for communication and, thus, lower the cognitive complexity needed for both when working on common artifact. We conjuncture, based on these experiences, that *common decoding experiences* are more important for formal modeling than establishing *common knowledge up front*.

Lastly, we note that the metaphors established by the formal model can transfer novel structures into the domain: For example, the notion of a logical group is used recently for infrastructure in railways by Schön (2021), but stems from its formal modeling as a common object-oriented pattern to group objects for communication (Kamburjan et al., 2018).

Objective Naturalness. Our notion of modeling and naturalness is subjective, relying on the inner workings of the mind, and we do not investigate *objective* naturalness, which would directly connect the semiotic symbol with the semiotic thing (Fig. 1). Indeed, it is questionable whether such a notion could exist. One can argue that domains have inherent structure, which should be natural to any model and modeler acquainted with the domain. This brings our discussion back to the above point about domain-specific languages, which aim to provide a natural model *for any mind*, and we stress that this is not the same as a natural model *without an involved mind*, which does not exist in our framework[10].

Empirical Evaluation of Naturalness. As such, any precise, objective assessment the naturalness would have to rely on direct measurement of cognitive complexity of individual artifact-person (or artifact-mind) pairs. That would in turn require direct access to mental representations, which is still beyond the present state of the art. On the other hand, indirect characterizations of naturalness across formalisms, artifacts and mind types might still be possible within the realm of experimental Cognitive Sciences. We let this issue for future work.

User Studies in Formal Methods. Formal methods, as well as related disciplines, rarely perform user studies that target understanding and tend to reuse theories from human-computer interaction. Consequently, they are restricted to usability questions. For example, Hentschel et al. (2016) propose a new tool for interactive theorem proving that is motivated by enabling the user to understand the formal system better:

> *"To improve the efficiency of understanding intermediate proof situations, therefore, promises considerable gains in the overall human user time spend..."*

However, their study is purely *performative* and only investigates whether the tool increases the performance with respect to time and correctness. It does not investigate whether the tool indeed improves understanding.

Similarly, Harkes (2018) discusses the problems when evaluating domain-specific languages, where the standard approach in that field is to discuss (1) performance and (2) generality, because these are simple to measure and simple to argue over. This problem of arguing about languages is particularly explicit in the work of Myers et al. (2004) on natural programming, which argues that programming languages and environments should be "natural":

> *"By natural, we mean faithfully representing nature or life"*

We regard this definition as little useful in practice, as it gives no detail to why a model is more natural than another and ignores that naturalness differs between individuals. Note, however, that Myers et al. are interested in program development and are *"aiming for the language and environment to work the way that nonprogrammers expect"*. They are not considering single models/programs.

While not a user study, the presentation of Leuschel (2017) is worth mentioning in this context: it argues that the reason why the B-method is so successful

[10] In the semiotic framework there is no such thing as a model *at all* without an involved mind, as a model is a sign and a sign needs an interpreter.

in modeling railway systems is that railway systems are modeled as graph structures and the B-method is well-suited to operate on such graph structures. We interpret this as a naturalness argument in our sense: given a B-model, it is easy to retrieve the domain view from the computational structures.

Further Related Work We are interested in formal models from the perspective of cognitive linguistic and largely ignore the actual evaluation or runtime semantics. Indeed, we do not require a computer or (runtime) semantics in the first place. Tanaka-Ishii (2009) gives a more detailed view on programs, where the symbols signify their semantics, which are, in turn, again signs. The sense-making process in that setting is not mental, but physical. Other works on semiotics in computing, such as the one by Andersen (1994), also focus on the relation of the sign to the execution itself. In contrast, there is a tradition of Semiotic Engineering in Human-Computer Interaction (HCI) (de Souza and Leitão, 2009) that sees computers as devices for communication, not computation and draws some parallels between the development of programming and the evolution of "natural" languages (Blackwell, 2017)[11]. This research has also resulted in some cognitive guideline for dimensions for *usability* of programming languages environments (Blackwell et al., 2001). From these dimensions *Closeness of mapping* comes closest to naturalness. Similarly, for business processing modeling languages understandability has been investigated by Fahland et al. (2009).

Colburn and Shute (2008) investigate metaphors for programming, which has a rich vocabulary of metaphors such as *thread* or *throw/catch*. They observe that for programming, the metaphors are, in the terms of Indurkhya (1992), better explained through rather *comparative* theories. Comparative theories of metaphors see metaphors as ways to emphasize and expose preexisting similarities between two domains. In contrast, we use an *interactive* theory of metaphors, where the metaphors creates the similarity. Furthermore, Colburn and Shute discuss metaphors in the opposite direction: While formal models are computer scientific structures that are metaphors for some domain, their metaphors are terms from some domain for computer scientific structures. In subsequent work, this approach is applied to types (Colburn and Shute, 2017). Metaphors are also used widely in HCI (Blackwell, 2006).

Another connection between the philosophy of science and formal modeling has been explored by Hähnle (2018), who notes that black boxes have, in general, a negative connotation in philosophy, as they prevent the investigation of its content, while they have, again in general, a positive connotation in computer science, because they hide complexity.

Works in in Ontology Engineering in Computer Science also touch in some of the notions we discussed here. Guarino (1998) proposed that ontologies specified as logical theories should approximate the set of intended models (i.e. first-order models) in the the domain, without investigating how the mismatch might occur. Guizzardi (2005) suggests a similar distinction between mental models

[11] On the problems of applying the theory of evolution to developments of programming languages we refer to Crafa (2015).

of domain concepts and artifacts, as well as their representation as symbolic specifications. Also, he summarizes a collection of mappings to characterize how well a formal model covers a domain. However, the work also does not investigate in more detail how artifact-specific constructs affect ontologies. Furthermore, some works in ontology also incorporate Conceptual Spaces as a representation construct (Guizzardi, 2005; Aisbett and Gibbon, 2001), however we go further in representing the formal artifact itself.

7 Conclusion

This article presents a cognitive view on formal modeling motivated by the observation that several effects in formal modeling that cannot be analyzed by focusing on abstraction, i.e., the reduction feature of models.

At the core are two proposals. (1) That there are two sense-making processes associated with a formal model, one that interprets the formal model as a concept in the application domain and one that interprets it as a concept in the engineering domain. We represent these mental concepts using conceptual spaces. (2) That a model is more natural than another, if it is a better metaphor, i.e., it retains more structure from the engineering view in the domain view.

The notion of naturalness is our main contribution: a formal model is more natural than another if it is cognitively easier to map the extracted artifact concept on the extracted domain concept. As naturalness is concerned with mental processes starting and ending with mental concepts, we also introduce *perceptional naturalness* as a notion of complexity to measure the difference between the artifact itself and the artifact concept it is decoded into. This captures a wide range of effects of more syntactical nature, e.g., formatting and naming.

Contrary to prior work, we do not focus on programming, i.e., execution, or user interfaces, but on a single aspect of formal modeling. We hope that a cognitive view on formal modeling can lead to better designed modeling languages, better designed qualitative user studies and help to build a body of experiences in formal modeling.

Future Work. The natural next step is to design user studies to empirically test our view. A promising start to do so is to investigate are common modeling experiences for interdisciplinary modeling efforts. Furthermore, as we are motivated by the difficulties of justifying and precisely argue about modeling decisions, we also plan to reinvestigate recent successful formal modeling projects, namely `FormbaR` (Kamburjan et al., 2018), the `GeoAssistant` (Din et al., 2019), and the core ontology for robotics and automations (IEEE ORA WG, 2015; Fiorini et al., 2017), in particular its positioning part (Carbonera et al., 2013), and present the underlying modeling decisions using the framework presented here.

We conjecture that investigating the connection with semiotics in more detail can give further insights into modeling. For example, the situation of the modeler can be seen as multiple parallel signifying processes (cf. Bateman (2018, Fig. 2)). Furthermore, the relation of $\mathfrak{M}^\delta$ and $\mathfrak{C}^\delta$ has similarities to the relation of the dynamic and final interpretant of Peirce (Chandler, 2017).

Acknowledgements This work was partially supported by the Research Council of Norway via the `SIRIUS` Center (237898) and the `PeTWIN` project (294600). The authors thank Lars Tveito and Michael Lienhardt for feedback on early drafts of this article.

Bibliography

Aisbett, J. and Gibbon, G. (2001). A general formulation of conceptual spaces as a meso level representation. *Artificial Intelligence*, 133(1-2):189–232.

Andersen, P. B. (1994). *A semiotic approach to programming*, page 16–67. Learning in Doing: Social, Cognitive and Computational Perspectives. Cambridge University Press.

Bateman, J. A. (2018). Peircean semiotics and multimodality: Towards a new synthesis. *Multimodal Communication*, 7(1):20170021.

Blackwell, A. F. (2006). The reification of metaphor as a design tool. *ACM Trans. Comput. Hum. Interact.*, 13(4):490–530.

Blackwell, A. F. (2017). *6,000 Years of Programming Language Design: A Meditation on Eco's Perfect Language*, pages 31–39. Springer.

Blackwell, A. F., Britton, C., Cox, A., Green, T. R. G., Gurr, C., Kadoda, G., Kutar, M. S., Loomes, M., Nehaniv, C. L., Petre, M., Roast, C., Roe, C., Wong, A., and Young, R. M. (2001). Cognitive dimensions of notations: Design tools for cognitive technology. In *Cognitive Technology: Instruments of Mind*, pages 325–341. Springer.

Carbonera, J. L., Fiorini, S. R., Prestes, E., Jorge, V. A. M., Abel, M., Madhavan, R., Locoro, A., Gonçalves, P. J. S., Haidegger, T., Barreto, M. E., and Schlenoff, C. (2013). Defining positioning in a core ontology for robotics. In *IEEE/RSJ*, pages 1867–1872. IEEE.

Chandler, D. (2017). *Semiotics: The Basics.* Routledge, 3rd edition.

Colburn, T. and Shute, G. (2008). Metaphor in computer science. *Journal of Applied Logic*, 6(4):526–533.

Colburn, T. R. and Shute, G. M. (2017). Type and metaphor for computer programmers. *Techné: Research in Philosophy and Technology*, 21:71–105.

Crafa, S. (2015). Modelling the evolution of programming languages. *CoRR*, abs/1510.04440.

de Souza, C. S. and Leitão, C. F. (2009). *Semiotic Engineering Methods for Scientific Research in HCI.* Synthesis Lectures on Human-Centered Informatics. Morgan & Claypool Publishers.

Din, C. C., Karlsen, L. H., Pene, I., Stahl, O., Yu, I. C., and Østerlie, T. (2019). Geological multi-scenario reasoning. In *32nd Norsk Informatikkonferanse, NIK.* Bibsys Open Journal Systems, Norway.

Fahland, D., Lübke, D., Mendling, J., Reijers, H. A., Weber, B., Weidlich, M., and Zugal, S. (2009). Declarative versus imperative process modeling languages: The issue of understandability. In *BMMDS/EMMSAD*, volume 29 of *Lecture Notes in Business Information Processing*, pages 353–366. Springer.

Fiorini, S. R. and Abel, M. (2013). Part-whole relations as products of metric spaces. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 55–62. IEEE.

Fiorini, S. R., Bermejo-Alonso, J., Gonçalves, P. J. S., de Freitas, E. P., Alarcos, A. O., Olszewska, J. I., Prestes, E., Schlenoff, C., Ragavan, S. V., Redfield, S. A., Spencer, B., and Li, H. (2017). A suite of ontologies for robotics and automation [industrial activities]. *IEEE Robotics Autom. Mag.*, 24(1):8–11.

Gärdenfors, P. (2004). *Conceptual spaces: The geometry of thought.* MIT press.

Gärdenfors, P. (2014). *The geometry of meaning: Semantics based on conceptual spaces.* MIT press.

Guarino, N. (1998). Formal ontologies and information systems. In *Formal Ontology in Information Systems, Proceedings of FOIS 98.* IOS Press.

Guizzardi, G. (2005). *Ontological foundations for structural conceptual models.* PhD thesis, University of Twente.

Hähnle, R. (2018). Colorful boxes. In *The 7th Biennial Conference of the Philosophy of Science in Practice*, pages 147–148. Faculty of Arts and Philosophy, University of Ghent.

Harkes, D. (2018). We should stop claiming generality in our domain-specific language papers. *The Art, Science, and Engineering of Programming*, 3.

Hentschel, M., Hähnle, R., and Bubel, R. (2016). An empirical evaluation of two user interfaces of an interactive program verifier. In *ASE*, pages 403–413. ACM.

IEEE ORA WG (2015). IEEE standard ontologies for robotics and automation. *IEEE Std 1872-2015*, pages 1–60.

Indurkhya, B. (1992). Metaphor and cognition. an interactionist approach. *Studies in Cognitive System.*

Isaac, A. M. C., Szymanik, J., and Verbrugge, R. (2014). Logic and complexity in cognitive science. In *Johan van Benthem on Logic and Information Dynamics*, pages 787–824. Springer.

Johnsen, E. B., Steffen, M., Stumpf, J. B., and Tveito, L. (2018). Resource-aware virtually timed ambients. In *IFM*, volume 11023 of *LNCS*, pages 194–213. Springer.

Kamburjan, E., Hähnle, R., and Schön, S. (2018). Formal modeling and analysis of railway operations with active objects. *Sci. Comput. Program.*, 166:167–193.

Kühne, T. (2006). Matters of (meta-)modeling. *Softw. Syst. Model.*, 5(4):369–385.

Lakoff, G. (1993). *The contemporary theory of metaphor*, page 202–251. Cambridge University Press, 2 edition.

Lakoff, G. and Johnson, M. (1980). *Metaphors we Live by.* University of Chicago Press.

Leuschel, M. (2017). The unreasonable effectiveness of B for data validation and modelling railway systems. RSSRail, Keynote.

Myers, B. A., Pane, J. F., and Ko, A. J. (2004). Natural programming languages and environments. *Commun. ACM*, 47(9):47–52.

Olmstead, B. (1998). Reference Malbolge interpreter. https://www.lscheffer.com/malbolge_interp.html, retrieved 29.10.2021.

Peirce, C. S. (1935). *The Collected Papers of Charles Sanders Peirce*. Harvard University Press.

Peled, D. A. (2001). *Software Reliability Methods*. Springer.

Quinlan, D., Wells, J. B., and Kamareddine, F. (2019). BNF-style notation as it is actually used. In *CICM*, volume 11617 of *Lecture Notes in Computer Science*, pages 187–204. Springer.

Rosch, E. and Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7(4):573–605.

Schön, S. (2021). Formalisierung von betrieblichen Regelwerken. In *SRSS'21 Tagungsband*. TU Darmstadt. In German.

Sivik, L. and Taft, C. (1994). Color naming: A mapping in the IMCS of common color terms. *Scandinavian journal of psychology*, 35(2):144–164.

Stachowiak, H. (1972). *Allgemeine Modelltheorie*. Springer. in German.

Steen, G. J. (2011). The contemporary theory of metaphor — now new and improved! *Review of Cognitive Linguistics*, 9(1):26–64.

Stehr, M. and Meseguer, J. (2004). Pure type systems in rewriting logic: Specifying typed higher-order languages in a first-order logical framework. In *Essays in Memory of Ole-Johan Dahl*, volume 2635 of *LNCS*, pages 334–375. Springer.

Tanaka-Ishii, K. (2009). *Semiotics of Programming*. Cambridge University Press.

Thorne, C. (2012). Studying the distribution of fragments of english using deep semantic annotation. In *8th Workshop in Semantic Annotation ISA 8*.

Ullmann, S. (1972). *Semantics: An Introduction to the Science of Meaning*. Basil Blackwell.

van Rooij, I. (2008). The tractable cognition thesis. *Cogn. Sci.*, 32(6):939–984.

Warglien, M. and Gärdenfors, P. (2013). Semantics, conceptual spaces, and the meeting of minds. *Synthese*, 190(12):2165–2193.

Wittgenstein, L. (1953). *Philosophical Investigations*. Basil Blackwell, Oxford.