2. Towards the Ontological Representation of Functional Basis in DOLCE

Stefano Borgo[1], Massimiliano Carrara[2], Pawel Garbacz[3], and Pieter Vermaas[4]

[1] Laboratory for Applied Ontology, ISTC-CNR, Italy
[2] Department of Philosophy, University of Padua, Italy
[3] Department of Philosophy, The John Paul II Catholic University of Lublin, Poland
[4] Department of Philosophy, Delft University of Technology, The Netherlands

Abstract. We present a formalisation of the notion of function as set forward by Robert Stone and Kristin Wood in the Functional Modelling approach. This formalisation is part of a larger project to analyse functional descriptions of technical artifacts. The current result is a formal system in which engineering notions of functions are expressed in terms of definitions grounded in the foundational ontology DOLCE. We identify those notions that are particularly significant from the point of view of ontology formalisation. Then we match them against the conceptual framework defined by. Finally, we posit a number of constraints to eliminate some unintended interpretations of our formalism.

Keywords: Function, Technical artifact, Foundational ontology, Engineering

1 Introduction

In this paper we present a formalisation of the notion of function of technical artifacts as set forward by Robert Stone and Kristin Wood (see for example [10]). This formalisation is part of a larger project to analyse functional descriptions of technical artifacts. In this project we follow a bottom-up methodology. We take the engineering literature as a starting point and formalize the main engineering notions of functions available in terms of formalized definitions grounded in the foundational ontology of DOLCE – the *Descriptive Ontology for Linguistic and Cognitive Engineering* [8]. The DOLCE ontology is part of the *WonderWeb* effort.[5] The vision of this effort is to have a library of foundational ontologies reflecting different ontological choices. For that reason extensive attention is given to a careful isolation of the ontological options and their formal relationships. DOLCE's ontology is the most studied module of this library. In [1] we have already formalised the notion of function and the related notion of behaviour as defined by Chandrasekaran and Josephson [4]. In this paper we continue our ontological analysis of the engineering domain by giving a first formalisation in DOLCE of one of the rival notions of function, which is endorsed by R. Stone and K. Wood. Our larger goal is to build a series of integrated formal systems and to compare these with the state of the art in ontology. Indeed, other groups

[5] http://wonderweb.semanticweb.org

have already presented more or less comprehensive ontological frameworks for these and similar topics, in particular Mizoguchi's group [7, 6] although the methodology is fairly different. Unfortunately, the ontological comparison of our results to other works requires at least an extended discussion of the different rendering of the engineering literature (let alone the basic ontological assumptions). This task is not simple and we aim to dedicate a future paper on this comparison.

The plan of the present paper as follows. In section 2 we introduce Stone and Wood's model and identify its key functional notions. In section 3 we briefly present DOLCE's framework and, in section 4, we give an ontological characterisation of this model. The concluding section discusses some limits of our approach, looks at other approaches and indicates future steps.

2 Functional Basis

Briefly put, the approach of Stone and Wood [10] propounds to model overall product functions, especially from the electromechanical and mechanical domain, as sets of connected elementary sub-functions. In line with the design methodology of [9] an overall product function is described by means of a verb-object form and graphically represented by a black-boxed operation on flows of materials[6], energies and signals. Sub-functions are also described by verb-object forms but they correspond to well-defined basic operations on well-defined basic flows of materials, energies and signals. Roughly, the black-boxed operations on general flows correspond product functions and are derived from costumer needs, the basic operations and basic flows correspond to sub-functions. All these notions are limited in number and stored in shared libraries. These libraries cover the functional design space. The whole system is called the Functional Basis (FB).[7]

Stone and Wood present their proposal as supporting the archiving, comparison and communication of functional descriptions of existing products, as well as the engineering designing of new products. Archiving, comparison and communication is assisted since the sub-functions into which overall product functions are decomposed, are described in this universal common language. Designing of new products is supported since the Functional Basis allows designers to make critical design decisions about the architecture of artifact by decomposing overall product functions in sub-functions in the early conceptual stage of designing at which only functional descriptions are considered. Moreover, this approach provides the means to find overall design solutions quickly since Stone and Wood require the sub-functions to be small and easily solvable in designing.

In the designing of new artifacts the overall product function is given in terms of input/output flows and may initially be coarse-grained. Consider the following example: the overall product function of loosening/tightening screws, which is performed or ascribed to a power screwdriver, is defined by several input flows: electricity, human force, on/off signals, and screw; and one output flow: looseness/tightness of screws. But when this overall

---

[6] The notion of material is construed here rather broadly as it comprises also such objects as screws, air, human beings and their parts.

[7] Sometimes, the overall engineering methodology related to FB is called Functional Modeling.

product function is decomposed in terms of connected sub-functions with well-defined input and output flows, the overall product function can be modeled in more detail. The input flows are extended to include also relative rotation, and the output to include also heat and noise.

In [10] Stone and Wood provide a glossary of the terms used in their account, which we resume here for later reference. Note that the term 'function' indicates an operation on a flow, which has the awkward consequence that neither product functions nor sub-functions (as described above) are instances of functions in the FB terminology.

- *Product function*: the general input/output relationship of a product having the purpose of performing an overall task, typically stated in verb-object form.
- *Sub-function*: a description of part of a product's overall task stated in verb-object form. and represented by an input/output flow relationship. Sub-functions are decomposed from the product functions and represent the more elementary tasks of the product.
- *Function*: a description of an operation to be performed by a device or artifact. It is expressed as the active verb of the sub-function.
- *Flow*: a change in material, energy or signal with respect to time. Expressed as the object of the sub-function, a flow is the recipient of the function's operation.

More in general:

- *Functional model*: a description of a product or process in terms of the elementary functions that are required to achieve its overall function or purpose.
- *Functional basis*: a design language consisting of a set of functions and a set of flows that are used to form a sub-function.

Earlier analysis of the advantages and drawbacks of the FB approach can be found in [5, 11].

3 DOLCE

DOLCE is a foundational ontology of particulars with a clear cognitive bias. In fact, its aim is to capture the ontological categories underlying natural language and human commonsense. For this reason the categories introduced in DOLCE are thought by its developers as "cognitive artifacts ultimately depending on human perception, cultural imprints and social conventions" ([8], p.13). The categories are obtained by the analysis of the surface structure of language and cognition. Consequences of this approach are that DOLCE's categories are at the so-called *mesoscopic level*, the level of the middle-sized (commonsensical) objects we, as humans, perceive. For DOLCE is not proposed as an ontology for the hard sciences like physics, rather it collects descriptive notions that assist in making explicit already formed conceptualizations by grounding them within the human cognitive perspective.

3.1 DOLCE: a short introduction

DOLCE's taxonomic structure is pictured in figure 1. Each node in the graph is a category

of the ontology. A category is a subcategory of another if the latter occurs higher in the graph and there is an edge between the two. PARTICULAR is the top category. The class of subcategories of a given category forms a partition except where dots are inserted.

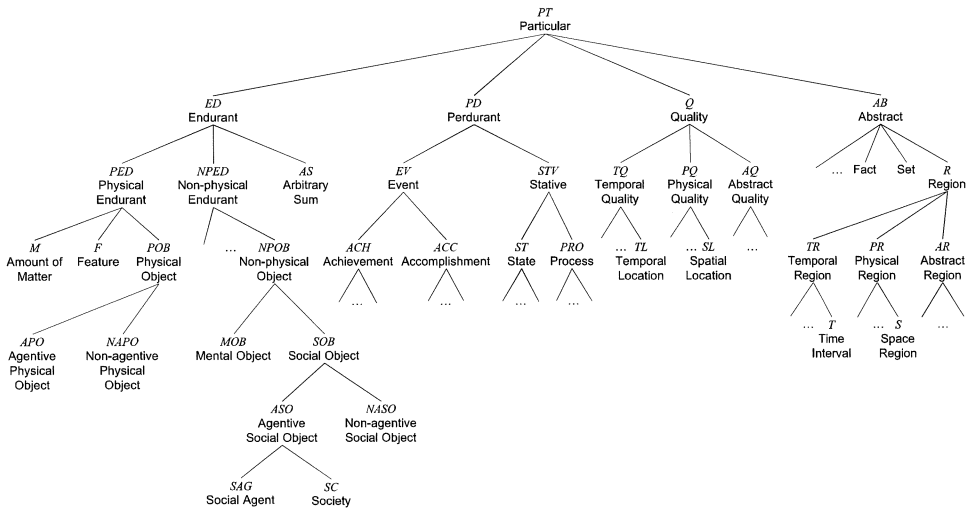

Fig. 1. DOLCE taxonomy

As said in the introduction, we want to extend DOLCE to capture crucial notions in the area of engineering design so to use this ontological framework to analyze, clarify and possibly improve the work in this area. Here we can provide just a minimal introduction to the whole ontology, the interested reader can find in [8] the underlying motivations and a throughout discussion of technical aspects.

From the graph in figure 1, it is clear that the DOLCE ontology concentrates on particulars as opposed to universals. Roughly speaking, a *universal* is an entity that is instantiated or concreted by other entities (like "being human" and "being an event"). A *particular* (an element of class PARTICULAR) is an entity that is not instantiated by other entities (like the Eiffel Tower in Paris). That is, your car is a particular as opposed to the model of your car, provided that the latter is interpreted as a type being instantiated by a number of entities among which one is your car. Particulars comprise physical or abstract objects, events, and qualities. It seems to us that the DOLCE ontology provides a good framework for the needs of engineering design: it adopts the distinction between objects like products and events like operations; it includes a differentiation among individual qualities (e.g., the weight of a specific material item), quality types (weight, color and the like), quality spaces (spaces to classify weights, colors etc.), and quality positions or qualia (informally, locations in quality spaces). These, together with measure spaces (where the quality positions get associated to a measure system and, thus, to numbers), are important to describe and compare devices. Indeed, among the motivations to use DOLCE, an important element was its robustness and flexibility which allows to capture in a natural way the views proper of engineering practice.

The DOLCE ontology category (class) ENDURANT comprises objects (like a *hammer*) or amounts of matter (like *an amount of plastic*), while the category PERDURANT comprises events like *making a hole* or *playing a soccer game*, that is, things that happen in time. (Formally, the category ENDURANT is represented by the constant ED and we write $\mathsf{ED}(x)$ to mean that x is an endurant. Analogously, we use PD for perdurants.) The term 'object' is used in the ontology to capture a notion of unity as suggested by the partition of the class PHYSICAL ENDURANT (formally, PED) into classes AMOUNT OF MATTER (M), FEATURE (F), and PHYSICAL OBJECTS (POB) (see figure 1). Both endurants and perdurants are associated with a bunch of individual qualities (elements of the category QUALITY). The exact list of qualities may depend on the entity: *shape* and *weight* are usually taken as qualities of endurants[8], *duration* and *direction* as qualities of perdurants. An individual quality, e.g., the weight of the car you are driving, is a quality associated with *one and only one* entity; it can be understood as the particular way in which that entity instantiates the general property "having weight". For example, the endurant Hammer_#321 (a token) has its own individual instantiation of property "having weight", namely, the individual weight-quality of Hammer_#321. The change of an endurant in time is explained through the change of some of its individual qualities. For example, with the substitution of a component, Hammer_#321 may change its weight. This means that the individual weight-quality of this entity was first associated to (or classified in) a position x and later to (in) a position y of a given weight-quality space. Note that x and y should not be considered weight measures like, say, $5kg$. They are elements of (positions in) a quality space whose primary role is to partition individual qualities in equivalent (or similar, depending on the space) entities before committing to numeric values and measure units. Thus, the same x may be associated to $5kg$ in one measure space and to $11.1lb$ in another. Quality spaces are elements of the REGION category, a subcategory of ABSTRACT.

Another important relation for our analysis is the *parthood* relation: "x is part of y", written: $\mathsf{P}(x,y)$, with its cognates the *proper part* (written: $\mathsf{PP}(x,y)$) and *overlap* relations (written: $\mathsf{O}(x,y)$), which may be defined in terms of P. In DOLCE the *parthood* relation applies to pairs of endurants (e.g., one endurant is part of another) as well as to pairs of perdurants (to state that an event is part of another). For pairs of endurants, the relation of parthood is temporalised (i.e. with the third argument for time objects) since an endurant may loose and gain parts throughout its existence.

While relations like parthood, overlap and sum are inherited from classical mereology theory and are fairly standard in formal ontology, other relations need a brief introduction. The main relation involving both endurants and perdurants is called *participation*, formally PC. This relation captures the simple fact that an endurant 'lives' in time by participating in some perdurant. For example, a machine (endurant) may participate in a production process (perdurant). A car's life is also a perdurant to which that car participates throughout all the duration (the time spanned from the construction of the car till its destruction). If endurant x participates in perdurant y at each instant of period t, we write $\mathsf{PC}(x,y,t)$ which reads "x participates in y during all of t" (here t may be just a part of the duration of y).

Note that participation and parthood are distinct relations. An endurant is never part of a perdurant, only perdurants can be parts of perdurants (analogously, only endurants

[8] In DOLCE they are called physical qualities and are denoted by means of the PQ predicate.

can be parts of endurants). Also, participation is time-indexed in order to account for the varieties of participation in time like temporary participation, constant participation, and so on, cf. [8]. Two important DOLCE relations are qt and ql. In DOLCE the first relates endurants, perdurants or qualities to their own qualities. That is, given a quality x and an endurant (perdurant, quality) y, relation $\mathsf{qt}(x, y)$ holds if x is an individual quality of y. To relate qualia (i.e. regions in a quality space) to qualities, possibly with a temporal parameter, relation ql is used. Expression $\mathsf{ql}(x, y)$ stands for "x is the quale of individual quality y" and $\mathsf{ql}(x, y, t)$ for "x is the quale of individual quality y at time t". In DOLCE ql is also used to define relations between, say, an endurant and its qualia. For instance, if T is the temporal quality space, we write $\mathsf{ql}_{\mathsf{T,ED}}(t, x)$ to mean that t is the temporal quale of endurant x.

We also make use of a couple of relations which are not part of DOLCE but of an extension proposed in [1]. The first relation, called *wholly participation* ($\mathsf{PC}_{\mathtt{WH}}$), is a specialisation of the participation relation. $\mathsf{PC}_{\mathtt{WH}}(x, y)$ stands for "endurant x wholly participates in the perdurant y", that is, x participates in y throughout all the time spanned by y.[9] The second relation identifies combinations of perdurants that may co-occur according to engineering science: two perdurants x and y are said to be *coherent*, written $\mathsf{Coh}(x, y)$, when their mereological sum [10] is a (physically) possible perdurant. This notion accounts for the fact that some combinations of perdurants are meaningless in engineering practice. It is possible that an air-conditioning system cools the room, at a given time, and it is also possible that the same system heats the room, at the same time. However, these two perdurants cannot 'belong to the content' of the real world, i.e. they are not coherent in the sense of Coh.[11]

4 From Functional Basis model to an ontology of engineering functions

Our task, the development of an ontology-based formalisation of the Functional Basis model, is carried out in a three-stage process. First, we identify the notions in the model that are particularly significant from the point of view of ontology formalisation (subsection 4.1 below). Then we match them against the conceptual framework defined by DOLCE (4.2). Finally, we posit a number of constraints to eliminate some unintended interpretations of our formalism (4.3).

Most of engineering models are not sufficiently clear with respect to the ontological categorisations of the notions they employ. Thus, there is some degree of arbitrariness when it comes to constructing ontological interpretations of those models. In this work we consider the models furnished by FB as models of product tokens (endurants) and characterize them by referring to perdurants. Note that in the design literature one easily mixes the token and the type levels of products (sometimes intertwining information about other types of enti-

[9] $\mathsf{PC}_{\mathtt{WH}}$ should not be confused with the DOLCE relation PC_{T} since a further condition is imposed on y, that is, y must belong to a class of generalized perdurant relevant to engineering practice, see [1]. This extra condition is not discussed here.

[10] The mereological sum of x and y is a perdurant z such that each part of x and each part of y are parts of z, and if any perdurant w overlaps z, then w also overlaps x or y.

[11] Cf. [1] for a formal definition.

ties like the blueprint) so that our choice may fail to capture some engineering views. This general problem may be overcome by the development of complementary perspectives to be integrated with the main ontological view we adopt. We do not discuss this issue further in the paper.

4.1 Ontological aspects of Functional Basis

In table 4.1 we collect some central notions that are later included in the formalisation either as primitives or as derived terms. These notions are crucial to successfully import FB in an ontology. The examples are taken from the functional model of power screwdriver constructed in [10].

Table 1. Primitive ontological notions in FB

Term	**Explanation**	**Example in FB**
Req(x, y, z, v)	engineering function x, device y, and flows z and v satisfy the design requirements	*loosen/tighten screws* power screwdriver electricity and human force satisfy the design requirements
EngFun(x, y, z)	x is an engineering function with input flow y and output flow z	*transmit torque* is an engineering function with torque as its input and output flow
Dev(x)	x is a device	power screwdriver is a device
InFlow(x, y)	x is an input flow for function y	electricity is an input flow for *convert electricity to torque*
OutFlow(x, y)	x is an output flow for function y	torque is an output flow for *convert electricity to torque*
MFlow(x)	x is a flow of matter	solid is a flow of matter
EFlow(x)	x is a flow of energy	electricity is a flow of energy
SFlow(x)	x is a flow of signal	on/off information is a flow of signal

The notion of design requirements needs a few comments since we use it as a catch-all for terms like 'requirement', 'customer need' and 'user demand'. Indeed, Stone and Wood do not set a definite notion for the variety of requirements FB needs to consider. They may use a variety of more or less general expressions like "requirements on human flows" or "customer needs for the product" [10]. Pahl and Beitz [9] are however of help in justifying an emphasis on the notion of design requirements as collecting user demands and other technical demands (idem, Chapter 5) and make clear how it serves the purpose of determining the functional structure in terms of flows and operations (idem, Chapter 6). The notion of design requirements thus seems to provide a bridge for designers between user demands and functions, and this explains why we emphasize its role in the Functional Modeling approach. The design requirements are supposed to model all conditions that constrain a particular design problem. These include the physical laws, customers' needs, legal regulations, company's policies, etc. These interrelated factors provide a design context in which a product being designed is ascribed certain functionalities. Following the Functional Basis' construal of engineering functions, these functionalities are considered in terms of operations on flows. This implies that the initial design requirements explicitly drive the selection and configuration of operations and flows.

Secondly, although different design situations involve different design requirements,

for the sake of formal simplicity we assume in our formalisation that the design situation is fixed. This is to say that the predicate Req as presented here cannot distinguish different design contexts.[12]

Thirdly, we allow complex flows as arguments of the Req relation. Complex flows can be formalized in different ways and the specific technical solution is not important here. The introduction of complex flows is crucial, though, since the (de)composition of functionalities, following the FB approach in the constrained situation given by the design requirements, involve most of the times multiple input and/or output flows of different types.

Functional Basis definitions We take Req, Dev and several notions of flow as primitives in our framework. Req is a complex relationship and deserves further study. Unfortunately, as mentioned earlier, it is left largely unexplored in the FB literature and we do not have much information to reach a definite formalisation. On the positive side, we can say that the whole framework does not depend on the specific choice one takes in modeling Req. Dev is here considered a subcategory of PHYSICAL OBJECT in the DOLCE ontology. It may be understood as a specialisation of the notion of ARTIFACT [3], a notion proposed as an extension of DOLCE, but this is debatable and we do not take a position in this paper. Indeed, the distinction between device, artifact and product in FB is not yet clear to us. These terms are not properly defined and their use is not clear from the text. Stone and Wood use expressions like "a product or device carries out [an operation]", "an operation to be performed by a device or artifact", and "the creation of an artifact, product, system, or process" [10] which hint to some distinction but are insufficient to establish if this has an ontological or contextual nature. The FB notions of flow, material (MFlow), energy (EFlow), and signal (SFlow), are fairly well discussed in the FB approach and in the next section we propose a clear ontological stand for them. Here we use them to define a single notion of flow that is more general than that in FB, the motivations are given below. Finally, two further notions of flow, namely input (InFlow) and output (OutFlow) flows, are context dependent: a flow can be an input flow for an operation and an output flow for another so these notions do not correspond to categories. Rather, they are contextual classifications that one may prefer to model as roles or as relational properties in the sense of [2]. Moreover, it seems that an input flow for an operation (i.e. function) is somehow involved in the generation of one of the output flows for this operation. This involvement may be represented by the notion of causality, but we do not attempt to tackle this difficult problem here.

We say that an event x is an engineering function with input flow y_1 and output flow y_2 if there is some device z which participates to the whole event x such that $\mathtt{Req}(x, z, y_1, y_2)$. Formally,

$$\begin{aligned}&\mathtt{EngFun}(x, y_1, y_2)\\ &\triangleq \exists z[\mathtt{Dev}(z) \wedge \mathtt{InFlow}(y_1, x) \wedge \mathtt{OutFlow}(y_2, x) \wedge \mathtt{PC}_{\mathtt{WH}}(z, x) \wedge \mathtt{Req}(x, z, y_1, y_2)]. \quad (1)\end{aligned}$$

[12] In the full formalisation, this can be achieved by introducing a further argument to Req. Although this is not particularly problematic here, this choice has important effects on the definition of EngFun given below.

We need to introduce an extra definition whose relevance will become clear shortly. An engineering function *simpliciter* is an event for which there exist input and output flows that satisfy definition (1). Formally,

$$\texttt{EngFun}(x) \triangleq \exists y_1, y_2 \texttt{EngFun}(x, y_1, y_2). \tag{2}$$

FB partition flows in three disjoint types: material, energy and signal (cf. [9], p. 29-30). Introducing predicates `MFlow`, `EFlow`, and `SFlow` for these types, we write

$$\texttt{Flow}(x) \triangleq \exists y, z, w(\texttt{MFlow}(y) \wedge \texttt{EFlow}(z) \wedge \texttt{SFlow}(w) \wedge x = y \oplus z \oplus w).^{13} \tag{3}$$

This classification gives ontological information since it depends on what the flow is *about*. In the perspective of function description, instead, we have seen that flows of the types listed can participate to functions as input or as output flows. In other terms, the function at stake provides the contextual perspective to classify flows as input, output or even irrelevant to a function. In terms of the previous definition, this amounts to say that, for a generic flow x, there exists a function (perdurant) y such that any subflow of x of the allowed types is either input or output for y. Formally,

$$\texttt{Flow}(x) \equiv \exists y(\texttt{EngFun}(y) \wedge \forall z[((\texttt{MFlow}(z) \vee \texttt{EFlow}(z) \vee \texttt{SFlow}(z)) \wedge \mathsf{P}(z, x)) \rightarrow (\texttt{InFlow}(z, y) \vee \texttt{OutFlow}(z, y))]). \tag{4}$$

If one uses $\texttt{Flow}_{\text{FB}}$ for a flow in the pure FB terminology, that is, either material, energy or signal, a *simple flow* so to speak, then it would be possible to write

$$\texttt{Flow}_{\text{FB}}(x) \triangleq \exists y(\texttt{EngFun}(y) \wedge \texttt{InFlow}(x, y) \vee \texttt{OutFlow}(x, y)). \tag{5}$$

4.2 Functional Basis and DOLCE

As anticipated in the introduction to section 4, the engineering functions listed in FB are here interpreted as types whose instances are perdurants of certain kinds.[14] That is to say, an FB function is taken as a class of (engineering possible) perdurants, e.g., *to distribute*

---

[13] As anticipated, the meaning of the $\oplus$ operator is not given in detail. The technical aspects related to the formal treatment of complex flows is not relevant to the rest of the formalisation although some aspects, like temporal co-occurrence, are important. Note also that the notion of flow captured in this definition (or, in other terms, the notion of complex flow) is more general then the notion one finds in the FB approach. This generalization is not strictly necessary and it is adopted because we believe it clarifies the basic dependences between functions and flows. See for instance condition (14).

[14] It needs to be emphasised that FB does not use the notion of behaviour and relates technical functions directly to technical devices. This modelling decision does not account for the fact that functions are ontologically dependent on agents' intentions. As a consequence, FB functions may conceptually collapse into simple behaviours and this possibility led to one of the objections to FB - see [6]. Since in this paper we only attempt to interpret ontologically FB, we do not elaborate on this observation.

is a class of perdurants in which a certain flow is broken up.[15] The classes of perdurants used by FB are not arbitrary and can be characterised at least to some extent. The proposed match between engineering functions and classes of perdurants allows us to provide a coherent and consistent framework for the three notions used in FB: function, sub-function, and product function. Given our assumption that an FB engineering function is the most basic in the ontological sense and is formalized by means of a class of perdurants, an FB sub-function is modeled as the subclass collecting all and only the perdurants that have as participants the flows requested in the engineering description of the sub-function.

Axioms 21, 22, and 23 below specify the ontological nature of this involvement for different types of flows. For example, the sub-function *to distribute liquid* is represented here as a (proper) subclass of the class corresponding to the *to distribute function* in which a certain amount of liquid is distributed.

Note that the current axiomatisation is confined to the level of *instances* of FB function types; these instances correspond to perdurants in the chosen ontological framework.

$$\mathtt{Dev}(x) \rightarrow \mathsf{POB}(x) \tag{6}$$

$$\mathtt{EngFun}(x) \rightarrow \mathsf{PD}(x) \tag{7}$$

Generally speaking, we give a characterisation of the FB view of a product in terms of the (class of) perdurants to which it participates. Not all the perdurants are considered. Just those that are engineering possible and that satisfy a series of conditions like customer needs, standard requirements, e.g., being compliant with given standards and law restrictions. We have implicitly formalized these conditions via the Req relation in the previous section. It is important to note that only part of these constraints are due to natural laws and engineering practice. This observation shows that FB models comprise an intentional component that the ontological framework should clarify and make explicit.

At this point, it is natural to conclude that our general notion of function, as formalized in (1), corresponds to the FB notion of sub-function. That is to say, a triple formed by a perdurant and two flows identifies an instance of an FB sub-function whenever the triple satisfies (1). Similarly, the generalized notion of function, defined in (2), corresponds to the FB notion of function. Finally, predicate Req identifies (instances of) FB product functions when the second argument (the argument constrained to devices) is occupied by an object that is considered in the respective design situation to be a product. In short, if we use Prod, $\mathtt{SubFun}_{\mathrm{FB}}$, $\mathtt{Fun}_{\mathrm{FB}}$, and $\mathtt{ProdFun}_{\mathrm{FB}}$ for, respectively, products, FB sub-functions, FB functions, and FB product functions, then we would have the following definitions:

$$\mathtt{SubFun}_{\mathrm{FB}}(x, y, z) \triangleq \mathtt{EngFun}(x, y, z) \tag{8}$$

[15] FB defines this function in the following way:

> Distribute. To cause a material or energy to break up. The individual bits are similar to each other and the undistributed flow. Example: An atomizer distributes (or sprays) hair-styling liquids over the head to hold the hair in the desired style. ([10], p. 368)

$$\mathtt{Fun}_{\mathrm{FB}}(x) \triangleq \mathtt{EngFun}(x) \tag{9}$$

and restricting x to products, i.e. if $\mathtt{Prod}(x)$ holds, then[16]

$$\mathtt{ProdFun}_{\mathrm{FB}}(x, y, t, v) \triangleq \mathtt{Req}(x, y, t, v) \tag{10}$$

The following three conditions make clear the ontological categories of FB flows. They state that material flows are (a subcategory of) endurants, energy flows are (a subcategory of) physical qualities of endurants and signal flows are (a subcategory of) perdurants.

$$\mathtt{MFlow}(x) \rightarrow \mathsf{ED}(x) \tag{11}$$

$$\mathtt{EFlow}(x) \rightarrow \mathsf{PQ}(x) \tag{12}$$

$$\mathtt{SFlow}(x) \rightarrow \mathsf{PD}(x) \tag{13}$$

These three conditions provide also the underpinning for the ontological basis of predicate `Flow`, see (3). We can thus constrain ontologically the arguments of the `Req` relation

$$\mathtt{Req}(x, y, z, v) \rightarrow \mathsf{PD}(x) \wedge \mathtt{Dev}(y) \wedge \mathtt{Flow}(z) \wedge \mathtt{Flow}(v) \tag{14}$$

4.3 Additional constraints

First, we want to tie the design requirement relation to the notion of engineering function. If relation `Req` holds for a function x, and entity y and flows z_1, z_2, then we require that `EngFun` holds for x, z_1 and z_2.

$$\mathtt{Req}(x, y, z_1, z_2) \rightarrow \mathtt{EngFun}(x, z_1, z_2). \tag{15}$$

In turn, if x is a device then there exists function and flows such that `Req` is satisfied.

$$\mathtt{Dev}(x) \rightarrow \exists y, z_1, z_2 \mathtt{Req}(y, x, z_1, z_2). \tag{16}$$

The possibility to combine functions is crucial for the Functional Basis approach. However, there is lack of explanations about the rules that govern function (de)composition. A rule for function composition can be obtained via the notion of coherent perdurants: roughly, two perdurants are coherent if their sum is a perdurant that is admissible within

---

[16] Obviously,

$$\mathtt{Prod}(x) \rightarrow \mathtt{Dev}(x)$$

However, the notion of product is relative to a design context, i.e. what is considered as a product in one design situation, might be considered as a part of another product in a different design situation. We do not fully characterise this notion here mainly because we do not represent different design requirements by means of `Req`. Once these requirements are admitted as the fifth argument of `Req`, we could relativise the notion of product to those requirements so that the context-relativity of products would be captured.

engineering science (see pg. 6).

$$\mathtt{Req}(x, y, z_1, z_2) \wedge \mathtt{Req}(x', y, z_1, z_2) \wedge \mathtt{Coh}(x, x') \rightarrow \mathtt{Req}(x + x', y, z_1, z_2). \tag{17}$$

The rational reconstruction of the notions used in FB leads us to the conclusion that on the one hand one should separate input and output flows from devices performing functions and on the other hand one should bind those flows to functions themselves. Both postulates are based on the examples to be found in FB, but due to their generality they extend its ontological assumptions. The former postulate is based on the "black-box" conceptualisation of engineering functions, where flows are separated from devices. The latter refers to the understanding of those functions as operations on flows, which implies some kind of ontological link between them.

The following formulas posit some limits to the contribution of flows to functions. These constraints are independent of the input or output status of the involved flow as shown by the disjunctive component in the antecedents.

Separation axioms No material flow for a function of a device overlaps with the device itself.

$$\mathtt{MFlow}(x) \wedge [\mathtt{Req}(y, z, x, v) \vee \mathtt{Req}(y, z, v, x)] \wedge \mathsf{ql}_{\mathsf{T,ED}}(t, x) \rightarrow \neg \mathsf{O}(x, z, t). \tag{18}$$

No energy flow for a function is a quality of a device performing this function.

$$\mathtt{EFlow}(x) \wedge [\mathtt{Req}(y, z, x, v) \vee \mathtt{Req}(y, z, v, x)] \rightarrow \neg \mathsf{qt}(x, z). \tag{19}$$

No signal flow for a function has the device performing this function as a participant.

$$\mathtt{SFlow}(x) \wedge [\mathtt{Req}(y, z, x, v) \vee \mathtt{Req}(y, z, v, x)] \rightarrow \neg \mathsf{PC}(z, x, t). \tag{20}$$

Binding axioms Each material flow for a function participates to the function at least in part.

$$\mathtt{MFlow}(x) \wedge [\mathtt{Req}(y, z, x, v) \vee \mathtt{Req}(y, z, v, x)] \rightarrow \exists t \mathsf{PC}(x, y, t). \tag{21}$$

Each flow of energy for a function is involved in the function via the (possibly partial) participation of its bearer:

$$\mathtt{EFlow}(x) \wedge [\mathtt{Req}(y, z, x, v) \vee \mathtt{Req}(y, z, v, x)] \rightarrow \exists w, t[\mathsf{PC}(w, y, t) \wedge \mathsf{qt}(x, w)]. \tag{22}$$

Each signal flow for a function is (a proper) part of the function.

$$\mathtt{SFlow}(x) \wedge [\mathtt{Req}(y, z, x, v) \vee \mathtt{Req}(y, z, v, x)] \rightarrow \mathsf{PP}(x, y). \tag{23}$$

5 A conclusion

Returning to our overall project and taking stock of our results, we have formalized in [1] the engineering notion of function as defined by B. Chandrasekaran and J Josephson [4] and in this paper the engineering notion of FB function as set forward by Stone and Wood [10]. As such we have covered the two general approaches in engineering to capture functions: in the first *Functional Representation* approach captured by Chandrasekaran and Josephson, engineers relate functions of devices to behaviours of devices, and then relate these behaviours to structural-physical descriptions of the devices; in the second *Functional Modeling* approach represented by the work of Stone and Wood, engineers model functions of devices in terms of inputs and outputs, and then relate these functions directly to structural-physical descriptions of devices. The first approach grants a pivotal conceptual role to behaviour, suggesting an ontological ordering: devices have their physical structure; this structure, in interaction with a physical environment, gives rise to the devices' behaviours; and these behaviours then determine the devices' functions. The second approach seems to side-step the notion of behaviour and relates the physical structure of devices and their functions directly.

More specifically, Stone and Woods represent their FB functions of devices and their components by the operations that those items perform on flows of material, energy and signals. These functions are then analysed in terms of basic operations on flows. In our proposal FB functions are interpreted as function types whose instances are perdurants of certain kinds: FB functions are taken as particular classes of perdurants. These classes of perdurants chosen are not arbitrary: they are engineering possible perdurants that satisfy customer needs as well as standard requirements. Here we propose a consistent framework for the three notions used in FB: function, sub-function, and product function.

Our project is far from finished. We submit our current results as consistent formalisations of the two engineering notions of functions in the foundational DOLCE ontology. Yet a number of further issues need to be addressed, defining the next steps of our project. First of all, the formalisations given in [1] and in this paper should be validated as useful in engineering. Second, having observed that there are two approaches in engineering towards understanding functions, the question arises how the two are related. For our project this question amounts to the task of relating the formalisations of the notion of function as given in [1] and as presented here. When these two steps are accomplished, we will be in the position to compare our results with more sophisticated bodies of work in the ontology of engineering. A well-developed and proven proposal on engineering functions to date is given by the work of Riichiro Mizoguchi and Yoshinobu Kitamura.[17] A comparison of the two formal systems is foreseen. It is also needed to understand, at least in part, the ontological alternatives that one can take in formalizing the engineering domain.

---

[17] http://www.ei.sanken.osaka-u.ac.jp/english/

References

1. S. Borgo, M. Carrara, P. Garbacz, and P. E. Vermaas. A Formal Ontological Perspective on the Behaviors and Functions of Technical Artifacts. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1):(to appear), 2009.
2. S. Borgo and C. Masolo. Foundational Choices in DOLCE. In *Handbook on Ontologies (2nd edition)*. Springer, (to appear).
3. S. Borgo and L. Vieu. Artifacts in Formal Ontology. In A. Meijers, editor, *Handbook of the Philosophy of the Technological Sciences*, volume Part 2: Artifact ontology and artifact epistemology. Elsevier (to appear), 2009.
4. B. Chandrasekaran and J. Josephson. Function in Device Representation. *Engineering with Computers*, 16(3/4):162–177, 2000.
5. P. Garbacz. Towards a standard taxonomy of artifact functions. *Applied Ontology*, 1(3-4):221-236, 2006.
6. Y. Kitamura, Y. Koji, and R. Mizoguchi. An ontological model of device function: industrial deployment and lessons learned. *Applied Ontology*, 1(3-4):237–262, 2006.
7. Y. Kitamura and R. Mizoguchi. Ontology-based systematization of functional knowledge. *Journal of Engineering Desing*, 15(4):327–351, 2004.
8. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. Ontology Library. WonderWeb Deliverable D18. Technical report, CNR, http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf, 2003.
9. G. Pahl, W. Beitz, J. Feldhusen, and K. Grote. *Engineering Design. A Systematic Approach.* Springer, 3rd edition, 2007.
10. R. Stone and K. Wood. Development of a functional basis for design. *Journal of Mechanical Design*, 122(4):359–370, 2000.
11. P. E. Vermaas. The Functional Modelling Account of Stone and Wood: some critical remarks. In *16th International Conference on Engineering Design (ICED 07)*, 2007.