Res Eng Design (2013) 24:19–32
DOI 10.1007/s00163-012-0133-1

ORIGINAL PAPER

On the formal impossibility of analysing subfunctions as parts of functions in design methodology

Pieter E. Vermaas

Received: 9 November 2010 / Revised: 28 March 2012 / Accepted: 29 March 2012 / Published online: 24 April 2012

Abstract In this paper, a proof is given that in design methods, the relation between technical functions and their subfunctions in functional descriptions of technical products cannot be analysed as a formal relation of parthood. This result holds for design methods in which transformations of flows of energy, material and signals are accepted as functions. First, two specific categories of such technical functions are modelled. Second, the composition relation by which ordered sets of these functions define other functions is characterised. Third, it is shown that this composition relation for technical functions does not meet the basic postulates of parthood relations as given by mereology, the theory of parthood. It still may be beneficial to designing to take subfunctions informally as the parts of the functions they compose. Yet, the proof shows that when functional descriptions are formalised for, for instance, the development of automated design reasoning tools or for incorporation in engineering ontologies, the composition relation for technical functions cannot unconditionally be taken as a parthood relation.

Keywords Technical functions · Parthood relations · Functional composition · Functional decomposition · Modelling of functions · Engineering ontologies

List of symbols

φ, ψ	A technical function
ϕ	A token function
Φ	A type function
a, b, c	Token flows of energy, material and signals
I	The set of input flows of a token function
L	The set of flows locked in by a set of token functions
O	The set of output flows of a token function
Comp($\phi_1,\ldots,\phi_n$)	The token function to which the n token functions $\phi_1,\ldots,\phi_n$ compose
P(φ, ψ)	Part-of relation: φ is part of ψ
PP(φ, ψ)	Proper-part-of relation: φ is proper part of ψ

P. E. Vermaas (✉)
Philosophy Department, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands
e-mail: p.e.vermaas@tudelft.nl
URL: http://tbm.tudelft.nl/index.php?id=32454&L=1

1 Introduction

Functional descriptions of products form a key element in many design methods. Specifying the technical functions of a product that is to be designed belongs to the first steps towards characterising the product. And decomposing functions in subfunctions is seen as a technique by which designers can explore design solutions without being too early committed to a specific physical layout of the product. Design methods define rules for these functional descriptions, stating how functions are understood and represented, and laying down how functions are decomposed. The paradigmatic example is the method by Pahl and Beitz (1996), in which technical functions are taken as transformations of flows of energy, material and signals, and in which functional decomposition is an analysis of overall functions in terms of webs of basic functions.

By their key role and by being relatively well defined, functional descriptions make good candidates for more formal descriptions and for automated reasoning. Technical functions are indeed incorporated in engineering ontologies (e.g. Kitamura et al. 2005; Garbacz 2006; Arp and Smith 2008; Borgo et al. 2009, 2011; Burek et al. 2009; Goel et al. 2009). And computer tools for the decomposition of functions in subfunctions are developed (e.g. Sridharan and Campbell 2005; Bryant et al. 2006).

This paper supports the formalisation of functional descriptions by providing elements for modelling technical functions, and by proving that the relation between functions and their subfunctions cannot formally be taken as one of parthood.

First, a modelling is given of functions of two specific categories. This modelling is secondly extended to the composition relation between subfunctions and functions. Yet, using this modelling, it is thirdly proved that this composition relation cannot be analysed as a formal parthood relation for functions. According to mereology, the formal theory of parthood, a relation should at least meet three postulates for being a parthood relation, and it is shown that the composition relation for functions fails to do so. Possibly, it may be beneficial in design methods to informally understand subfunctions as parts of the functions they compose. This understanding may, for instance, be helpful because it suggests to designers to consider the design solutions to subfunctions as parts of the solutions to the overall functions. Providing propulsion is typically a subfunction of the overall function of many vehicles, and the engines that solve this subfunction are typically physical parts of the vehicles. The proof shows, however, that when functional descriptions in design methods are formalised for engineering ontologies or for automated reasoning tools, the composition relation for technical functions cannot be taken as a parthood relation.

A challenge to any general argument about functional descriptions is that there is not one unambiguous concept of function available in engineering (e.g. Erden et al. 2008; Vermaas 2009b; Van Eck 2010). Design methods define different rules for understanding and representing functions, and therefore, advance different procedures for functional decomposition. In the proof given in this paper, this ambiguity is not accommodated by considering the different engineering concepts of function separately; generality is achieved by considering functions of two specific categories that are arguably instances of functions on many design methods. The flip side of this strategy towards generality is that the functions considered in the proof are somewhat simple from an engineering point of view.

The results presented in this paper hold in this way for design methods in which transformations of flows of energy, material and signals are accepted as functions (e.g. Hubka and Eder 1988; Keuneke 1991; Lind 1994; Pahl and Beitz 1996; Sasajima et al. 1996; Modarres and Cheon 1999; Stone and Wood 2000; Chakrabarti and Bligh 2001; Otto and Wood 2001; Fantoni et al. 2009). And the results hold for methods in which this representation of functions is not adopted but for which it can be argued that such transformations of flows still count as special cases of functions (e.g. Umeda et al. 1996; Chandrasekaran and Josephson 2000; Goel et al. 2009).

In Sect. 2, the relation between functions and subfunctions is introduced, and in Sect. 3, it is discussed what it means to take this relation as a parthood relation. In Sect. 4, the three basic mereological postulates for parthood relations are given. Section 5 introduces the specific categories of functions considered in the proof, and in Sect. 6, the generality of this proof is analysed. Section 7 is about the composition of the considered functions. Section 8 defines parthood relations for these functions, which are measured against the mereological postulates in Sects. 9 and 10. Sections 11 and 12 conclude by assessing the proof and by discussing consequences for design methodology.

2 Functions and subfunctions

The relation between functions and subfunctions in functional descriptions of technical products is in design methods introduced by the technique of functional decomposition. Briefly put, and abstracting from differences between methods, it is a relation between one function φ and an ordered set of other functions φ_1, φ_2,…, φ_n, for which holds that the functions φ_1, φ_2,…, φ_n in the given ordering compose to φ. All the functions φ_i, with i running from 1 to n, then count as subfunctions of the composed function φ. Decomposition is not unique: there may exist two or more different sets of ordered functions that compose to φ. But composition is unique: a specific ordered set of function φ_1, φ_2,…, φ_n composes to just one function φ.[1]

When looking at examples of functional decomposition, the differences between methods surface, illustrating the lack of consensus in engineering how to understand and represent functions. The paradigmatic method is the one by Pahl and Beitz (1996), in which technical functions are taken as transformations of flows of energy, material and signals, and in which functional decomposition is an analysis of an overall function in terms of a web of basic functions. A simple example (1996, p. 33) is the

[1] One may challenge this last conclusion if one allows that an ordered set of functions φ_1, φ_2,…, φ_n that composes to a function φ also composes to any coarse-grained version of φ (e.g. Burek et al. 2009).

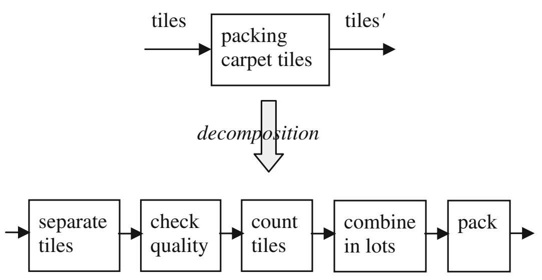


Fig. 1 A decomposition of the function of packing carpet tiles

decomposition of the function of packing carpet tiles in a series of sequentially ordered functions, see Fig. 1.

In addition to sequential orderings, subfunctions may also be ordered in parallel or in combinations of sequential and parallel orderings. Stone and Wood (2000), while still understanding functions as transformations of flows, have developed algorithms for such functional decompositions. An example in their approach is a decomposition of the function of holding liquid and retaining heat (Bryant et al. 2006, fig. 1), where now technical functions can also involve transformations of hands of users; see Fig. 2.

In both these examples, the ordering of the subfunctions is explicitly represented by the flows that are output of one subfunction and input to another. An alternative representation is adopted in the on-going work by Kitamura and Mizoguchi on functions and functional decomposition in the field of engineering ontologies. In that work, functional decompositions are captured by 'reverse tree'-like diagrams in which the ordering of subfunctions is left implicit. An example thereof is given in Fig. 3, which is the decomposition of the function of combining sheets of paper (a fragment adopted from Ookubo et al. 2007, fig. 3), which also illustrates that functional decompositions can be nested. In this example, individual decompositions are labelled by *ways of achievement*'s, represented by the little black squares interlinking the different functions (Kitamura and Mizoguchi 2003, sec. 4). I return at the end of this paper to this addition to the description of functional decompositions when discussing in Sect. 11 ways to circumvent the proof that the relation between subfunctions and functions cannot formally be a parthood relation.

3 A parthood relation for functions

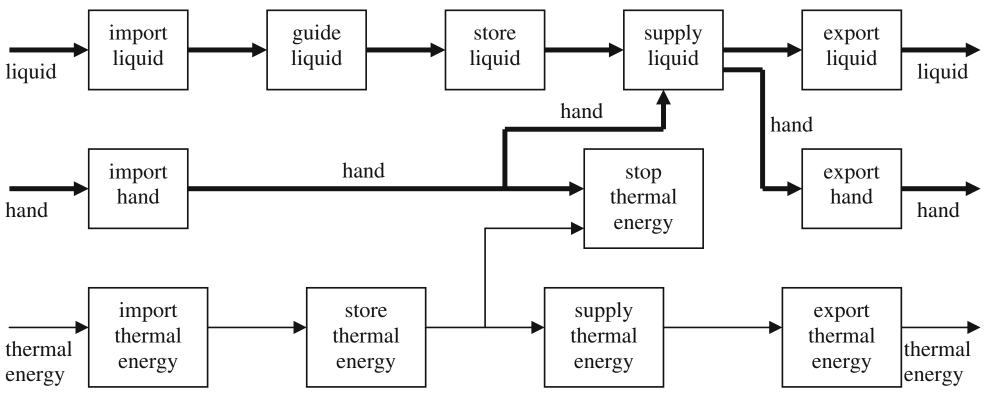
In the literature, the composition relation for functions is sometimes taken as a parthood relation. This characterisation can be found in design methodology (e.g. Lind 1994, p. 261; Umeda et al. 1996, p. 278), in engineering ontologies (e.g. Burek et al. 2009, p. 436), and in work that covers both these domains (e.g. Kitamura et al. 2004, p. 116). In design methodology, this characterisation may be taken in a colloquial sense, yet in engineering ontologies, the relation of parthood is a fundamental one that is described by mereology, the formal theory of parthood. This theory gives three basic postulates for any formal parthood relation, which are described in the next section. And although it is acknowledged that colloquial parthood relations need not meet these mereological postulates (Keet and Artale 2008), they are all three accepted to hold for formal parthood relations as considered in engineering ontologies. The proof given in this paper now demonstrates that these basic mereological postulates are violated by the composition relation for functions.

It should be emphasised that the parthood relation that is considered here is one between functions only. Existing work in engineering ontologies on parthood in relation to technical functions has been dominated by analyses of whether functional descriptions of products define a parthood relation for products and their components.[2] The parts and the wholes are then themselves not functions, but structural parts and wholes of products, or temporal parts and whole of processes associated with products. For instance, a functional description of a house may single out a door as a functionally defined structural part of the house, and a functional description of a door may single out a handle as a functionally defined structural part of the door. Central questions in this work are whether these functionally defined structural or temporal parthood relations meet the postulates of mereology, and how to explain possible violations. That such violations occur is generally accepted in mereology. It is, for instance, assumed that the basic mereological postulate of transitivity (see Eq. 3 in the next section) is typically not met, as is illustrated in the literature with the house-door-handle example: although the door is a functionally defined structural part of the house and the handle is a functionally defined structural part of the door, it is argued that the handle need not be taken as a functionally defined structural part of the house (e.g. Johansson 2004).

In this paper, the parthood relation considered is one directly at the level of function, that is, the part is a function and the whole is a function. So, in terms of the example, in this paper, part-of relations are considered between the function of the house, the function of the door and the function of the handle. This functional parthood relation is more rarely analysed in the literature but is finding its way to engineering ontologies (Kitamura et al. 2005; Burek et al. 2009), with some initial indications that

[2] A selection of this literature is given by: Winston et al. (1987), Simons and Dement (1996), Johansson (2004, 2005), Varzi (2005), Johansson et al. (2005), Vieu (2005), Vieu and Aurnague (2005) and Garbacz (2007).

Fig. 2 A decomposition of the function of holding liquid and retaining heat



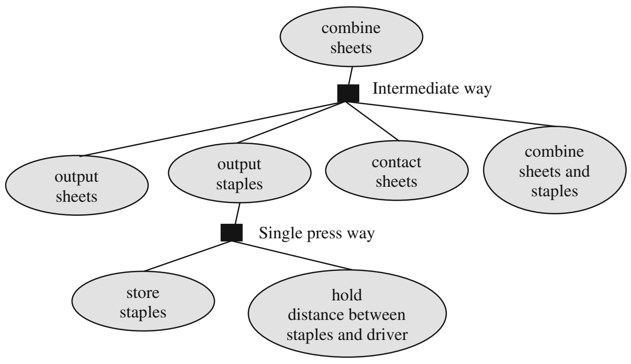


Fig. 3 A decomposition of the function of combining sheets

also it may not satisfy the basic postulates of mereology (Vermaas 2009a, 2010; Vermaas and Garbacz 2009).

4 Mereology and ground mereology

Mereology (Simons 1987; Casati and Varzi 1999; Varzi 2011), the theory of parthood, does not advance one relation of parthood. It rather defines and analyses different formalised parthood relations, which are compared with parthood relations as used in natural and specialised languages. Mereology thus provides a spectrum of possible parthood relations (Keet and Artale 2008), for physical objects, for temporal periods, for social entities, and so on, similar to how logic provides a spectrum of possible logics for different cases. Yet, in mereology, it is assumed that there is a common core to all parthood relations. This common core is called *ground mereology* and captured by simple postulates as given below. This assumption is normative: well-defined formalised parthood relations, as used in engineering ontologies, have to meet these postulates or are meeting them by construction; and when parthood relations as advanced in natural languages are not meeting the postulates, this is considered to be problematic and in need of explanation or mending (as in, e.g. Winston et al. 1987; Johansson 2004, 2005; Varzi 2005).

There are in mereology two distinct concepts of part, and the postulates of ground mereology depend on the specific concept adopted. The first is that of *proper part* by which an entity is by definition not a (proper) part of itself. The second concept is just called *part* and with this concept an entity is by definition always a part of itself. One can adopt as primitive either the concept of *proper part* or the concept of *part* and then define the other by means of the primitive one (see Eq. 5). The first concept of *proper part* comes closer to the everyday or engineering concept of part: it seems wrong to take a car as a part of itself, but taking the car's engine as a part of the car seems right, assuming the car does not consist of only the engine. The second concept of *part* in turn deviates from the everyday or engineering concept of part: a car is by this second concept a part of itself. It has, however, advantages to nevertheless choose this second concept of *part* as primitive, since parthood relations then define logically partial orderings (Varzi 2011), and in this paper, I adopt this choice. And as said, when taking this second concept of *part* as primitive, the first concept of *proper part* can be defined (Eq. 5) to recover the everyday or engineering concept of part.

Let P(φ, φ') represent a part-of relation, which is to be read as 'φ is a part of φ'', and which thus means that φ is identical to φ' or that φ is a proper part of φ'. The parthood relation made up by such part-of relations P(φ, φ') counts as a ground mereology if it meets three postulates: *reflexivity*, Eq. 1, *antisymmetry*, Eq. 2, and *transitivity*, Eq. 3:

$$\mathrm{P}(\varphi, \varphi) \tag{1}$$

$$(\mathrm{P}(\varphi, \varphi') \wedge \mathrm{P}(\varphi', \varphi)) \Rightarrow \varphi = \varphi' \tag{2}$$

$$(\mathrm{P}(\varphi, \varphi') \wedge \mathrm{P}(\varphi', \varphi'')) \Rightarrow \mathrm{P}(\varphi, \varphi'') \tag{3}$$

The postulate of reflexivity is, as discussed, somewhat odd when measured against the way in which the concept of part is used in everyday language and in engineering, but unproblematic given that the second concept of *part* is chosen as primitive. Antisymmetry is in mereology taken as unproblematic as well and sometimes even used to define identity between entities (e.g. Simons 1987, p. 50). The postulate of transitivity is, however, object of controversy, as discussed in the previous section.

Mereology has more postulates, and parthood relations are categorised by the additional postulates they meet. Here, I focus on the above minimal postulates of ground mereology but briefly mention the more special extensional parthood relations. These relations satisfy a fourth postulate called *strong supplementation* (Simons 1987; Casati and Varzi 1999; Varzi 2011) and meet the extensionality condition:

$$(\exists\psi(\mathrm{PP}(\psi,\varphi)) \vee \exists\psi(\mathrm{PP}(\psi,\varphi'))) \Rightarrow (\forall\psi(\mathrm{PP}(\psi,\varphi) \Leftrightarrow \mathrm{PP}(\psi,\varphi')) \Rightarrow \varphi = \varphi') \tag{4}$$

where PP(φ, φ') is the proper-part-of relation defined as:

$$\mathrm{PP}(\varphi,\varphi') = (\mathrm{P}(\varphi,\varphi') \wedge \neg\mathrm{P}(\varphi',\varphi)) \tag{5}$$

Condition Eq. 4 expresses that two entities φ and φ' are the same when they have exactly the same set of proper parts.

5 Modelling token and type functions

The proof that the relation between subfunctions and functions cannot be formally one of parthood consists of showing that this relation does not meet the three basic mereological postulates Eqs. 1–3. For this proof, two categories of functions are introduced. The first category consists of *token functions* ϕ modelled as transformations of tokens of flows, say, flows of energy, material or signals. With a token, flow is meant a specific flow that occurs only once. A token flow of electrical energy a, for instance, is a flow of energy that exists within one specific period in time and at one specific place in space. A second token flow of electrical energy a' may be a flow similar to a by having the same intensity and duration as a. But when this second flow a' occurs in another period or at another place than a, then a' is a token flow different to a. A token function ϕ can now be represented by ordered sets $\langle I, O\rangle$, where I contains the input token flows to the function ϕ and O its output token flows. For instance, the token function $\phi = \langle a, b\rangle$ transforms the electrical energy token flow a to the rotational energy token flow b. The token function $\phi' = \langle a', b'\rangle$, which transforms the electrical energy token flow a' to the rotational energy token flow b', is then another token function than $\phi = \langle a, b\rangle$ as soon as a and a' are two different token flows, or b and b' are two different token flows.

In engineering, technical functions are typically not token functions: if a and a' are two similar token flows and b and b' are also two similar token flows, the token functions $\phi = \langle a, b\rangle$ and $\phi' = \langle a', b'\rangle$ are typically taken as describing one and the same function. For being able to express this generalisation, a second category of functions is introduced: that of *type functions* Φ. Define types of flows X as equivalence classes $\{x, x', x'',\ldots\}$ of token flows that are taken to be similar in engineering, say, because they have the same content, intensity and duration. Two token functions $\phi = \langle I, O\rangle$ and $\phi' = \langle I', O'\rangle$ can then be taken as similar if their input flows in I and I' are pair-wise of the same types of flows and if their output flows in O and O' are pair-wise of the same types of flows. So, define type functions Φ as equivalence classes $\{\phi, \phi', \phi'',\ldots\}$ of similar token functions $\phi = \langle I, O\rangle$, $\phi' = \langle I', O'\rangle$, and so on. A type function can then be represented by $\Phi = \{\langle I, O\rangle,\langle I', O'\rangle,\ldots\}$. Thus, if the token flows of electrical energy a and a' are instances of the same type of flows, and if the token flows of rotational energy b and b' are instances of the same type of flows, one has that $\phi = \langle a, b\rangle$ and $\phi' = \langle a', b'\rangle$ are instances of the same type function, namely the type function Φ represented by $\{\langle a, b\rangle,\langle a', b'\rangle,\ldots\}$.

Type functions come closer to technical functions that are regularly considered in engineering yet are still rather specific. Technical functions are typically defined as operations with variable input, say, electrical currents within a certain range of amperage. Type functions are, however, referring to only transformations that have input flows of one specific type, say input electrical currents of 2 ampere only.

Token and type functions can be represented by arrows for flows and boxes for transformations, see Fig. 4. For token functions, the possibility of this representation is straightforward; for type functions, it has to be kept in mind that type functions are represented by token functions that are instances of the type.

6 The generality of the proof

The choice to focus in this paper on token and type functions is motivated by two considerations. The first is that the composition of token and type functions can be modelled easily, as is done in the next section. The second is that token and type functions are specific enough to be

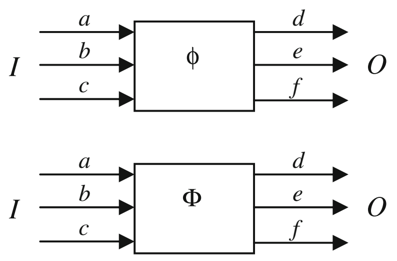


Fig. 4 Representations of token and type functions

instances of functions on many design methods. This implies that if it is shown that the composition relation for token and type functions cannot formally be taken as a parthood relation, it is also shown that in design methods that accept token and type functions, the composition relation for functions cannot formally be taken as a parthood relation.

As acknowledged in the introduction, design methods, or accounts of function, advance different concepts of technical function. A number of these methods and accounts accept the arrow-box representation of functions as given in Fig. 4 (e.g. Hubka and Eder 1988; Keuneke 1991; Lind 1994; Pahl and Beitz 1996; Sasajima et al. 1996; Modarres and Cheon 1999; Stone and Wood 2000; Chakrabarti and Bligh 2001; Otto and Wood 2001; Fantoni et al. 2009). Hence, token and type functions may be taken as instances of functions in these design methods and accounts, and it follows that the result of the proof holds for each of them.

A complication may be that some of these design methods and accounts impose additional constraints on functions. A first example is given by Modarres and Cheon (1999), who explicitly require that the representations of technical functions satisfy conservation laws as given by the natural sciences: the input flows listed in representations as given in Fig. 4 should together have equal energy as the listed output flows have, should in a similar way conserve charge, and so on. For instance, a function by which electrical energy is transformed to rotational energy with an efficiency of less than 1 should by Modarres and Cheon always be taken as a function that transforms electrical energy to rotational energy *and* to, say, thermal energy. Keuneke (1991) and Lind (1994) accept, in contrast, also functions in which the input and output flows do not match in terms of energy or of any other conserved quantity.[3] For Lind, for instance, creation of energy is an acceptable description of, say, the function of a battery.

A second example of additional constraints is given by design methods in which it is assumed that technical functions are composed of basic functions consisting of basic transformations of basic flows (e.g. Pahl and Beitz 1996; Stone and Wood 2000) as defined by libraries as the one given in (Hirtz et al. 2002). In order to let the proof given in this paper hold for all methods and accounts that represent functions by transformations of flows, I assume that the token functions $\phi = \langle I, O\rangle$ that figure in the proof (and thus the type functions as well) meet conservation laws and consist of basic transformations of basic flows. The token functions figuring in the proof actually consist of simple transformations of basic flows to other basic flows.

The proof may also hold for design methods in which the representation of functions by transformations of flows is not adopted (e.g. Umeda et al. 1996; Chandrasekaran and Josephson 2000; Goel et al. 2009). The functions considered in the proof may be accepted as instances of functions in these methods, or with some additions turned into instances of functions in the methods. Chandrasekaran (2005), for instance, does not take functions represented as transformations of flows as proper functions but nevertheless takes transformations as flows as giving information about functions as defined in the method by Chandrasekaran and Josephson (2000). By this perspective, the token and type functions considered in the proof can be interpreted as functions in the method by Chandrasekaran and Josephson (2000), such that the proof holds *mutatis mutandis* for this method as well. A similar attempt can be made to let the proof hold for the method by Goel et al. (2009). In this method, functions are represented by preconditions and postconditions, including references to the behaviour that accomplish the transitions from pre- to post-conditions. One can now argue that the pre- and post-conditions may consist of the input flows and the output flows that are transformed, and that then, when adding a reference to the behaviour that realises the transformation of these flows, the proof given in this paper can be made to apply to Goel's method also.

7 Composition of token functions

Because token functions are modelled as transformations of token flows, one has a straightforward way to also model their ordering and composition. To start with the ordering: the token flows that are shared by token functions represent the connections between these functions and in this way lay down their ordering.

Consider two token functions $\phi_1 = \langle I_1, O_1\rangle$ and $\phi_2 = \langle I_2, O_2\rangle$. Because the flows are tokens, the representations $\phi_1 = \langle I_1, O_1\rangle$ and $\phi_2 = \langle I_2, O_2\rangle$ of the individual functions already contain all information about how ϕ_1 and ϕ_2 are ordered. There are three cases:

(i) ϕ_1 and ϕ_2 are ordered in parallel if no token flows are shared by ϕ_1 and ϕ_2;

(ii) ϕ_1 and ϕ_2 are ordered in series if there are token flows in O_1 that are also in I_2 *or*[4] if there are flows in O_2 that are also in I_1;

[3] This tolerance does not imply that the technical phenomena that realise the functions violate conservation laws; the tolerance is merely that these laws may be ignored in representations of functions.

[4] This is an exclusive *or*.

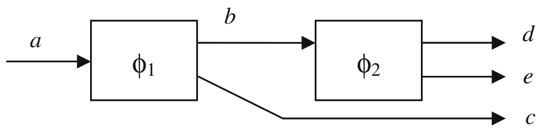


Fig. 5 An ordering and composition of two token functions

(iii) ϕ_1 and ϕ_2 are ordered in a loop if there are flows in O_1 that are also in I_2 *and* if there are flows in O_2 that are also in I_1.

For instance, the two token functions $\phi_1 = \langle a, \{b, c\}\rangle$ and $\phi_2 = \langle b, \{d, e\}\rangle$ are connected in series because the flow b is output of ϕ_1 and input to ϕ_2. In Fig. 5, this ordering is depicted by an arrow-box representation. Moreover, with this ordering, the functions ϕ_1 and ϕ_2 amount to a net overall transformation of the token flow a to the token flows c, d and e. Hence, the token function ϕ to which ϕ_1 and ϕ_2 compose is easily identified as the function represented by $\langle a, \{c, d, e\}\rangle$.

The analysis is still at a point of defining how to model composition of token functions. A connection with engineering can be made by taking the composition of the token functions $\phi_1 = \langle a, \{b, c\}\rangle$ and $\phi_2 = \langle b, \{d, e\}\rangle$ to the token function $\phi = \langle a, \{c, d, e\}\rangle$ as a simple-minded functional model of a combustion engine, with a a token flow of chemical energy, b a token flow of translational energy, d a token flow of rotational energy, and c and e two different token flows of thermal energy.

Before generalising this approach to the composition of n functions, I return for a moment to the modelling of a single token function $\phi_i = \langle I_i, O_i\rangle$. For an individual token flow b participating in this token function, one can discern three cases:

(i) flow b is in I_i only;
(ii) flow b is in O_i only;
(iii) flow b is in both I_i and O_i.

Case (iii) amounts to the flow b being a feedback loop as in $\phi_i = \langle\{a, b\}, \{b, c\}\rangle$, see Fig. 6.[5] In engineering, such feedback loops are typically accepted in the modelling of single functions: if a single token function has other flows a, c,… that are only in I_i or only in O_i, as in Fig. 6, the function still has a net input, net output, or both, and the feedback loop may be a means for controlling the output.

Formally one can also accept the further special case that all flows involved in a single token function amount to feedback loops, as in $\langle a, a\rangle$ and $\langle\{k, l\}, \{k, l\}\rangle$. Such token functions have no net input and no net output. Whether

[5] The aspect of time is not included in the modelling of token functions. This aspect can be added by specifying when token flows occur, and then it should be noted that flows that form feedback loops, like the flow b in Fig. 6, become input to the functions concerned at later moments in time than that they are output.

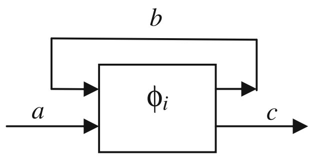


Fig. 6 A feedback loop in a single token function

such token functions make sense from an engineering point of view may be a point of discussion. The token function of an ideal waste dumping site may be taken as such a function: all output flows are reabsorbed by being input flows as well. And a product that is fully sustainable may be taken as having a token function in which all input flows are made up by all output flows. Yet, one can also argue that for being of practical use a product should have a token function that minimally absorbs a flow, say acoustic energy for a sound barrier, or that minimally creates a flow, say electrical energy for a battery.

One can introduce a condition on the modelling of single token functions that rules out functions with no net input and no net output. Let L be the set of all token flows that are feedback loops of a function ϕ_i:

$$L = I_i \cap O_i \tag{6}$$

(Below, Eq. 10, when again considering functional composition, I define a generalisation of this set L as all flows that are 'locked in' by a set of token functions; for a single token function, L is the set of flows locked in by that function.) A condition that avoids functions with no net input and no net output is then:

$$I_i \cup O_i / L \neq \varnothing \tag{7}$$

Yet, even when accepting this condition, token functions with no net input and no net output can resurface when functional composition is modelled, as will become clear below.

Impossible cases in the modelling of single token functions are given by flows b that occur more than once in I_i or more than once in O_i: a token flow b cannot enter or leave a transformation twice or more often. These cases are avoided in the modelling by taking I_i and O_i as sets. A consequence of this choice is that the splitting and merging of flows has to be modelled as transformations of different token flow, as in $\langle a, \{a', a''\}\rangle$ and $\langle\{a', a''\}, a\rangle$, while expressing separately that a, a' and a'' are, say, all token flows of electrical energy, all flows of water or all flows of signals.

Let us return to the modelling of functional composition and consider now n token function $\phi_1 = \langle I_1, O_1\rangle$, $\phi_2 = \langle I_2, O_2\rangle$,…, $\phi_n = \langle I_n, O_n\rangle$. Let Comp($\phi_1$, ϕ_2,…, ϕ_n) be the token function to which these n token functions compose. Because the functions $\{\phi_1, \phi_2,…, \phi_n\}$ are transformations of token flows, there is again a limited number of cases for the occurrence of a specific token flow b:

(i) flow b is in one input set I_i of the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$;
(ii) flow b is in one output set O_j of the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$;
(iii) flow b is in one output set O_i *and* in one input set I_i of the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$;
(iv) flow b is in one output set O_i *and* in one input set I_j of the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$ with $i \neq j$.

What is impossible is that a token flow b occurs in two or more different input sets I_i, I_j,…, or in two or more different output sets O_i, O_j,…; a token flow cannot simultaneously be input to two different functions or simultaneously be output of two different functions. Conditions on sets $\{\phi_1, \phi_2,\ldots, \phi_n\}$ of token functions that compose another token function are therefore:

$$I_i \cap I_j = \varnothing \quad \text{with} \quad i,j = 1,\ldots,n; i \neq j \tag{8}$$

$$O_i \cap O_j = \varnothing \quad \text{with} \quad i,j = 1,\ldots,n; i \neq j \tag{9}$$

In case (i), the flow b represents a flow that is only input to one of the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$ and thus also input to the composition Comp($\phi_1, \phi_2,\ldots, \phi_n$). In case (ii), the flow b represents a flow that is only output of one of the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$ and thus also output of the composition Comp($\phi_1, \phi_2,\ldots, \phi_n$). In case (iii), the flow b represents a feedback loop as depicted in Fig. 6 for one of the functions in $\{\phi_1, \phi_2,\ldots, \phi_n\}$, and in case (iv), the flow b represents a connection, as depicted in Fig. 5, between two different functions in $\{\phi_1, \phi_2,\ldots, \phi_n\}$. Flows that represent loops, that is, case (iii), are flows that are 'locked in' by the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$ and so are flows that represent connections, that is, case (iv). And as locked-in flows, they are neither input to nor output of Comp($\phi_1, \phi_2,\ldots, \phi_n$). The input flows to Comp($\phi_1, \phi_2,\ldots, \phi_n$) can now be identified as all input flows to the individual functions in $\{\phi_1, \phi_2,\ldots, \phi_n\}$ save those flows that are locked in by the functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$. And in a similar way, the output flows of Comp($\phi_1, \phi_2,\ldots, \phi_n$) can be identified.

Let L be again the set of locked-in flows. For a set of token functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$, L is given by the union:

$$L = \cup_{i,j=1,\ldots,n} I_i \cap O_j \tag{10}$$

Hence, the composite function Comp($\phi_1, \phi_2,\ldots, \phi_n$) is represented by:

$$\mathrm{Comp}(\phi_1, \phi_2, \ldots, \phi_n) = \langle \cup_{i=1,\ldots,n} I_i/L, \cup_{i=1,\ldots,n} O_i/L \rangle \tag{11}$$

It may happen that all flows participating in a set of functions $\{\phi_1, \phi_2,\ldots, \phi_n\}$ are locked in, creating again the special case of a token function with no net input and no net output. Consider, for instance, the composition of the token functions $\phi_1 = \langle\{a, b\}, \{c, d\}\rangle$, $\phi_2 = \langle\{c, d\}, \{a, e\}\rangle$ and $\phi_3 = \langle e, b\rangle$. The composite function Comp(ϕ_1, ϕ_2, ϕ_3) is then given by $\langle\varnothing, \varnothing\rangle$, see Fig. 7.

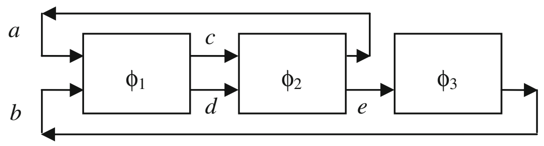


Fig. 7 A composite token function with only locked-in flows

Such composite token functions with no net input and no net output are again from a formal point of view possible outcomes of composition. And such composite functions still arise when one adopts condition Eq. 7 for avoiding that the single token functions $\phi_1, \phi_2,\ldots, \phi_n$ in the composition have no net input and no net output. So, if one also wants to rule out composite token functions with no net input and no net output, a generalisation of condition Eq. 7 should be accepted:

$$\cup_{i=1,\ldots,n} I_i \cup O_i/L \neq \varnothing \tag{12}$$

Composition of type functions is not defined in this paper[6]; the parthood relation for token functions is introduced by means of the composition of token functions, and the parthood relation for type functions is introduced as a generalisation of the parthood relation for token functions.

8 Parthood relations for token and type functions

Assume that the relation between token subfunctions and the token functions they compose defines a parthood relation P(ϕ', ϕ) for token functions as generated by the following sufficient condition:

$$\phi = \mathrm{Comp}(\phi_1, \ldots, \phi_n) \Rightarrow \forall\phi'(\phi' \in \{\phi_1, \ldots, \phi_n\} \Rightarrow \mathrm{P}(\phi', \phi)) \tag{13}$$

The part-of relations obtained by this condition describe actual states of affairs. That is, P(ϕ', ϕ) holds only if the token function ϕ' is *actually* a subfunction of the token function ϕ; it does not imply that ϕ' is a part of ϕ if ϕ' is *possibly* though not actually, a subfunction of ϕ. So, a specific token function of transforming electrical energy to

[6] Attempts to define composition of type functions via composition of token functions will in general not fix one particular composite type function. For instance, a composition Comp(Φ_1, Φ_2) with $\Phi_1 = \{\langle a, b\rangle, \langle a', b'\rangle,\ldots\}$ and $\Phi_2 = \{\langle b, c\rangle, \langle b', c'\rangle,\ldots\}$ may via the token compositions Comp($\langle a, b\rangle$, $\langle b, c\rangle$) = $\langle a, c\rangle$ and Comp($\langle a, b\rangle$, $\langle b', c'\rangle$) = $\langle\{a, b'\}, \{b, c'\}\rangle$ lead to the identification of at least two composite type functions Comp(Φ_1, Φ_2) = $\{\langle a, c\rangle, \langle a', c'\rangle,\ldots\}$ and Comp(Φ_1, Φ_2) = $\{\langle\{a, b'\}, \{b, c'\}\rangle, \langle\{a'', b'''\}, \{b'', c'''\}\rangle,\ldots\}$.

rotational energy may be a part of another specific function of transporting people. But this specific part-of relation need not hold for other pairs of similar token functions. This generalisation is also in principle not necessary: there are clearly cases where transforming electrical energy to rotational energy is actually not a subfunction of the function of transporting people. When a parthood relation P(Φ′, Φ) is introduced for type functions, this generalisation is, however, made: a part-of relation P(ϕ', ϕ) that holds for two token functions because ϕ' is actually a subfunction of ϕ is then extended to other pairs of similar token functions. This generalisation may be plausible for some functions: the function of transforming electrical energy to rotational energy seems to always have conducting electrical energy as a subfunction. Moreover, though this generalisation may not be necessary for all cases, it may nevertheless be possible without contradiction. Hence, it makes sense to just try it. An advantage of having this generalisation would be, for instance, that one collects for designers all potential (though not actual) compositions and decompositions of functions by means of part-of relations between type functions.

Assume therefore that functional composition for token functions amounts also to a parthood relation for type functions, and assume that this second parthood relation is defined by the sufficient condition Eq. 13 and by the following sufficient condition for part-of relations P(Φ′, Φ) for type functions:

$$(\mathrm{P}(\phi', \phi) \wedge \phi' \text{ is of type } \Phi' \wedge \phi \text{ is of type } \Phi) \Rightarrow \mathrm{P}(\Phi', \Phi) \tag{14}$$

When it is assumed that the conditions Eqs. 13 and 14 define parthood relations, they cannot be necessary conditions. As parthood relations, they have to meet the three postulates Eqs. 1–3 of ground mereology. Two of these postulates, reflexivity, Eq. 1, and transitivity, Eq. 3, are generating additional part-of relations, as is illustrated below, and this implies that Eqs. 13 and 14 are merely sufficient conditions. The remaining postulate, antisymmetry, Eq. 2, puts in turn constraints on the set of part-of relations thus generated.

Before introducing cases in which the defined parthood relations for token and type functions run into trouble, it can be shown that for two classes of regular cases of functional composition and decomposition, the postulates of ground mereology are satisfied. The first class consists of cases in which a single token function ϕ is related to one set {ϕ_1,..., ϕ_n} of token subfunctions, say as in the packing-carpet-tiles example by Pahl and Beitz (see Fig. 1). Condition Eq. 13 leads in this case to the following part-of relations:

$$\mathrm{P}(\phi_i, \phi) \quad \text{for all} \quad i = 1, \ldots, n \tag{15}$$

The postulates of ground mereology add via reflexivity, Eq. 1, the relations:

$$\mathrm{P}(\phi, \phi) \text{ and } \mathrm{P}(\phi_i, \phi_i) \quad \text{for all} \quad i = 1, \ldots, n \tag{16}$$

Transitivity, Eq. 3, does not add further part-of relations: there are no non-trivial sequences P(ϕ, ϕ') and P(ϕ', ϕ''), and for the cases that $\phi = \phi'$ or $\phi' = \phi''$, the part-of relations P(ϕ, ϕ'') that the transitivity postulate adds are already given by Eqs. 15 and 16. Antisymmetry, Eq. 2, is met because for each non-trivial part-of relation P(ϕ_i, ϕ), there is no reverse part-of relation P(ϕ, ϕ_i). So, say, the token function of counting carpet tiles in Fig. 1 can formally be taken as a part of the composite token function of packing those tiles.

The second class consists of cases in which a single type function Φ is related to one set {Φ_1, Φ_2,..., Φ_m} of type subfunctions; the packing-carpet-tiles example can again be the illustration. It has to be assumed that there are no other compositions in which these type functions Φ or {Φ_1, Φ_2,..., Φ_m} are figuring because violations of ground mereology can occur when part-of relations originating from two different compositions of type functions are combined (see Sect. 10). Consider a single token composition ϕ = Comp(ϕ_1,..., ϕ_n) and let {Φ_1, Φ_2,..., Φ_m} be the set of different type functions that occur in the set {ϕ_1, ϕ_2,..., ϕ_n} (so one has $m \leq n$). There are now two options. The first is that ϕ is of a type Φ that is different to the types in {Φ_1, Φ_2,..., Φ_m}. Application of condition Eq. 14 then leads to the following part-of relations:

$$\mathrm{P}(\Phi_j, \Phi) \quad \text{for all} \quad j = 1, \ldots, m \tag{17}$$

$$\mathrm{P}(\Phi, \Phi) \text{ and } \mathrm{P}(\Phi_j, \Phi_j) \quad \text{for all} \quad j = 1, \ldots, m \tag{18}$$

The second option is that ϕ is of a type that is not different to the types in {Φ_1, Φ_2,..., Φ_m}. If ϕ is, for instance, of type Φ_k, the part-of relations become:

$$\mathrm{P}(\Phi_j, \Phi_k) \quad \text{for all} \quad j = 1, \ldots, m, \text{with} \quad j \neq k \tag{19}$$

$$\mathrm{P}(\Phi_j, \Phi_j) \quad \text{for all} \quad j = 1, \ldots, m \tag{20}$$

For both options all postulates of ground mereology are met.

9 Some initial negative results

The above results about parthood relations between functions are positive, but unfortunately they are also the only positive ones to report. Checking these parthood relations against additional postulates of mereology or for cases in which one has more than one composition or decomposition quickly yields negative results.

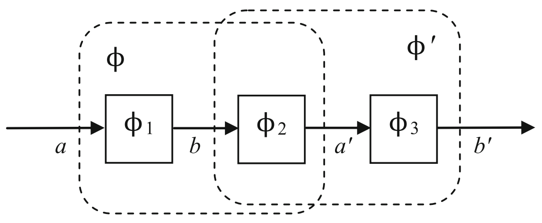


Fig. 8 ϕ_1 is part of ϕ, ϕ_2 is part of ϕ and of ϕ', and ϕ_3 is part of ϕ'

A parthood relation that is stronger than a ground mereology is the already mentioned extensional mereology that satisfy a fourth postulate called strong supplementation. Such an extensional parthood relation meets the following condition phrased in terms of proper-part-of relations:

$$(\exists\psi(\mathrm{PP}(\psi,\varphi)) \vee \exists\psi(\mathrm{PP}(\psi,\varphi'))) \Rightarrow (\forall\psi(\mathrm{PP}(\psi,\varphi) \Leftrightarrow \mathrm{PP}(\psi,\varphi')) \Rightarrow \varphi = \varphi') \tag{4}$$

This condition expresses that two functions that have the same proper parts are the same function. Token functions may satisfy this condition, yet type functions do not.

Consider the following case of three token functions $\phi_1 = \langle a, b\rangle$, $\phi_2 = \langle b, a'\rangle$ and $\phi_3 = \langle a', b'\rangle$, where a and a' are instances of the same type of flows and where b and b' are instances of the same type of flows. The first two functions compose to the token function $\phi = \mathrm{Comp}(\phi_1, \phi_2) = \langle a, a'\rangle$ and the last two compose to $\phi' = \mathrm{Comp}(\phi_2, \phi_3) = \langle b, b'\rangle$, as is illustrated in Fig. 8. With condition Eq. 13, one obtains for token functions the part-of relations:

$$\mathrm{P}(\phi_1,\phi),\ \mathrm{P}(\phi_2,\phi),\ \mathrm{P}(\phi_2,\phi') \text{ and } \mathrm{P}(\phi_3,\phi') \tag{21}$$

Since the token functions ϕ_1 and ϕ_3 are of the same type $\Phi_1 = \{\langle a, b\rangle,\langle a', b'\rangle,\ldots\}$, one obtains for type functions the part-of relations:

$$\mathrm{P}(\Phi_1,\Phi),\ \mathrm{P}(\Phi_2,\Phi),\ \mathrm{P}(\Phi_2,\Phi') \text{ and } \mathrm{P}(\Phi_1,\Phi') \tag{22}$$

with $\Phi_2 = \{\langle b, a'\rangle,\ldots\}$, $\Phi = \{\langle a, a'\rangle,\ldots\}$ and $\Phi' = \{\langle b, b'\rangle,\ldots\}$. In terms of proper-part-of relations, one obtains:

$$\mathrm{PP}(\Phi_1,\Phi),\ \mathrm{PP}(\Phi_2,\Phi),\ \mathrm{PP}(\Phi_2,\Phi') \text{ and } \mathrm{PP}(\Phi_1,\Phi') \tag{23}$$

The first proper-part-of relation PP(Φ_1, Φ), for instance, is obtained with Eq. 5 because one can derive for this particular case that P(Φ, Φ_1) does not hold: one does not have that $\phi = \langle a, a'\rangle$ is a part of $\phi_1 = \langle a, b\rangle$ or that $\phi = \langle a, a'\rangle$ is a part of $\phi_3 = \langle a', b'\rangle$; so, one does not have P(ϕ, ϕ_1) or P(ϕ, ϕ_3) for token functions; hence, one does not have P(Φ, Φ_1) for type functions. The other proper-part-of relations in Eq. 23 can be derived by similar reasoning, and together they prove that the parthood relation for type functions may violate extensionality, Eq. 4: the type functions $\Phi = \{\langle a, a'\rangle,\ldots\}$ and $\Phi' = \{\langle b, b'\rangle,\ldots\}$ have the same proper parts but are still different type functions.

This result is not surprising since it shows that the ordering of functions in a functional composition matters. Consider two type functions, the first Φ_1 being the increase of the temperature of a flow of material with 150° centigrade, and the second Φ_2 being the decrease of the temperature of a flow of material with 150° centigrade. These type functions can be composed by, intuitively, connecting them in series.[7] One has, however, two options: Φ_1 and then Φ_2 yields the composite type function Φ of baking the material flow; Φ_2 and then Φ_1 yields the type function Φ' of refrigerating that flow (Vermaas and Garbacz 2009).

A second result for the parthood relation for type functions generated by single compositions $\phi = \mathrm{Comp}(\phi_1,\ldots, \phi_n)$ is more counterintuitive. This result is that a simple and basic type function can have parts that are in general not in a sensible way related to the original type function.

Consider a case with the token flows a, b, c and d, and the token functions $\phi_1 = \langle a, b\rangle$, $\phi_2 = \langle b, c\rangle$ and $\phi_3 = \langle c, d\rangle$. Let a be a flow of electrical energy, b a flow of thermal energy, c a flow of chemical energy and d a flow of rotational energy. The token function $\phi_1 = \langle a, b\rangle$ then transforms the token flow a of electrical energy to the token flow b of thermal energy. This function ϕ_1 is a basic function in (Hirtz et al. 2002), and the same holds for ϕ_2 and ϕ_3. The functions ϕ_1, ϕ_2 and ϕ_3 compose to the token function $\phi = \mathrm{Comp}(\phi_1, \phi_2, \phi_3) = \langle a, d\rangle$, which transforms the token flow a of electrical energy to the token flow d of rotational energy. Condition Eq. 13 gives the following part-of relation for token functions:

$$\mathrm{P}(\phi_2,\phi) = \mathrm{P}(\langle b,c\rangle,\langle a,d\rangle) \tag{24}$$

In this case, the token flow a is indeed actually transformed to the token flow d via the intermediate transformation of flow b to c. Hence, the relation P(ϕ_2, ϕ) makes sense. But when this part-of relation P(ϕ_2, ϕ) is generalised to type functions, the result makes less sense. With condition Eq. 14, one obtains:

$$\mathrm{P}(\Phi_2,\Phi) = \mathrm{P}(\{\langle b,c\rangle,\ldots\},\{\langle a,d\rangle,\ldots\}) \tag{25}$$

which means that the type function Φ of transforming electrical energy flows to rotational energy flows has as a part the type function Φ_2 of transforming thermal energy flows to chemical energy flows. Abstracting from the example, the result is thus that *any* basic type function $\Phi = \{\langle a, d\rangle,\ldots\}$ can have many other basic type functions $\Phi_2 = \{\langle b, c\rangle,\ldots\}$ as its parts, where the token flows b and c are flows of types different to the types of flows a and

[7] Formally the connection between Φ_1 and Φ_2 is modelled by token functions that are instances of Φ_1 and Φ_2 and that have the right shared token flows.

d are instances of. This result stands perpendicular to the view that *some* of those basic type functions are truly basic in the sense of that they are not to be taken as decomposable into other functions (as in Hirtz et al. 2002).

10 Violating ground mereology

The last result derived in the previous section can be used to prove that the parthood relation for type functions violates ground mereology when part-of relations generated by two compositions of type function are combined (in Sect. 8, when considering the case of a single composition of type functions, this combination was avoided). By this result, the type function $\Phi_2 = \{\langle b, c\rangle, \ldots\}$ of transforming thermal energy flows to chemical energy flows may in turn be taken as having as a part the type function Φ of transforming electrical energy flows to rotational energy flows, that is:

$$P(\Phi, \Phi_2) \tag{26}$$

Together with the part-of relation $P(\Phi_2, \Phi)$ given in Eq. 25, this leads to a violation of the postulate of antisymmetry, Eq. 2: this postulate, applied to Eqs. 25 and 26, requires that Φ_2 is identical to Φ, which is not the case.

Equations 25 and 26 can also be derived by means of a single case. Consider again the second case described in the previous section with $\phi_1 = \langle a, b\rangle$, $\phi_2 = \langle b, c\rangle$ and $\phi_3 = \langle c, d\rangle$. But now assume that ϕ_2 is a composite of three other token functions $\phi_4 = \langle b, a'\rangle$, $\phi_5 = \langle a', d'\rangle$ and $\phi_6 = \langle d', c\rangle$, as in Fig. 9. Let the flows *a* and *a′* be flows of electrical energy of the same type, let *b* be a flow of thermal energy, let *c* be a flow of chemical energy, and let *d* and *d′* be flows of rotational energy of the same type. The

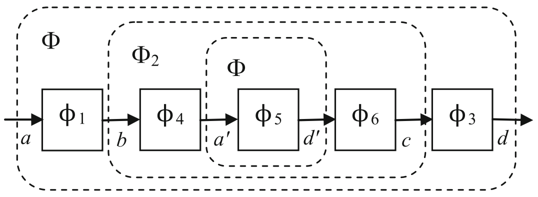


Fig. 9 Φ is part of Φ_2 and Φ_2 is part of Φ

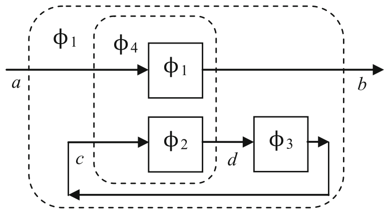


Fig. 10 ϕ_1 is part of ϕ_4 and ϕ_4 is part of ϕ_1

composition $\mathrm{Comp}(\phi_4, \phi_5, \phi_6) = \langle b, c\rangle = \phi_2$ gives the token part-of relation:

$$P(\phi_5, \phi_2) = P(\langle a', d'\rangle, \langle b, c\rangle) \tag{27}$$

Since ϕ_5 is a token function of type $\Phi = \{\langle a, d\rangle, \langle a', d'\rangle, \ldots\}$ and since ϕ_2 is a token function of type $\Phi_2 = \{\langle b, c\rangle, \ldots\}$, one obtains the type part-of relation $P(\Phi, \Phi_2)$, as in Eq. 26. The composition $\mathrm{Comp}(\phi_1, \phi_2, \phi_3) = \langle a, d\rangle = \phi$ again gives the token part-of relation $P(\phi_2, \phi)$, as in Eq. 24, by which one obtains the type part-of relation $P(\Phi_2, \Phi)$, as in Eq. 25.

The parthood relation for token functions can also violate ground mereology. This can be shown by considering combinations of token functional compositions simultaneously. Take a case with three token functions $\phi_1 = \langle a, b\rangle$, $\phi_2 = \langle c, d\rangle$ and $\phi_3 = \langle d, c\rangle$, where the first two functions compose to the token function $\phi_4 = \mathrm{Comp}(\phi_1, \phi_2) = \langle \{a, c\}, \{b, d\}\rangle$, and where this composite token function and the function ϕ_3 compose to $\mathrm{Comp}(\phi_3, \phi_4) = \langle a, b\rangle = \phi_1$, see Fig. 10. With condition Eq. 13, one obtains:

$$P(\phi_1, \phi_4),\ P(\phi_2, \phi_4),\ P(\phi_3, \phi_1) \text{ and } P(\phi_4, \phi_1) \tag{28}$$

For letting this set of relations meet the postulates of reflexivity, Eq. 1, and transitivity, Eq. 3, of ground mereology, additional parthood relations have to be added. Yet already with the four relations given in Eq. 28, a violation of the antisymmetry postulate, Eq. 2, can be detected: given $P(\phi_1, \phi_4)$ and $P(\phi_4, \phi_1)$, this postulate requires that $\phi_1 = \phi_4$, which is not the case.

An engineering example of this final case can be a functional model of a chemical process in a vessel transforming a liquid *a* to a liquid *b*. When pumping the liquid *a* into the vessel, a stream of air *d* has to leave the vessel, to be collected elsewhere before it streams as *c* back to the vessel when the liquid *b* is pumped out of the vessel.

In terms of the arrow-box representations given in Figs. 9 and 10, the violations of ground mereology by the parthood relations for token and type functions may be interpreted as showing that complex composites of arrows and boxes do not necessarily represent complex functions. A parthood relation for composites of arrows and boxes does not provide a parthood relation for the functions they represent.

A last observation is that in the proof that the parthood relation for token functions violates ground mereology, two functions ϕ_2 and ϕ_3 are considered that, together, compose to a function with no net input and no net output. This composite $\mathrm{Comp}(\phi_2, \phi_3)$ is not explicitly considered in the proof, and when composing functions by hand or by automated reasoning algorithms, such subsets of functions that compose to no-input-and-no-output components may occur and are formally possible. Conditions Eqs. 7 and 12, for instance, do not rule out the case represented in Fig. 10.

11 Assessing the proof

For assessing the proof that the composition relation for technical functions cannot be taken as a formal parthood relation, I recap its structure. Mereology provides reflexivity, antisymmetry and transitivity as postulates any formal parthood relation has to meet. For two categories of functions—token functions and type functions—the composition relations are spelled out, and it is then shown that they do not meet these mereological postulates. The functions in these categories are specific enough to be instances of functions on many design methods, in particular for methods in which functions are taken as transformations of flows. The result of the proof is in this way general: it holds for all types of formal parthood relations since all types of formal parthood relations have to meet the postulates of reflexivity, antisymmetry and transitivity; and it holds for all design methods that accept token and type functions as instances of technical functions.

The generality of the proof is achieved at the expense of the realism of the modelling of technical functions: the token and type functions considered in this paper and the cases used in the proof are not representative to typical engineering functional descriptions; they may be taken as 'toy examples'. This observation allows for challenging the result by requiring a proof that is based on more realistic functional descriptions. Such a challenge is reasonable from an engineering point of view, and not easy to meet, for instance, because it requires the modelling of functional composition for other categories of technical functions than token and type functions.

Yet, it is not feasible to challenge the proof by maintaining that realistic functional descriptions in engineering concern only single functional compositions or single decompositions. First, in engineering, more complex cases are used (see, e.g. Pahl and Beitz 1996, fig. 2.3; Ookubo et al. 2007, fig. 3, as partly reproduced in Fig. 3 in this paper). Second, even if only single decompositions are considered, as seems the rule in the method of Stone and Wood (2000), then combining part-of relations for type functions originating from different single decompositions is sufficient for disproving that the parthood relation for type functions meets the postulates of mereology, as was shown in the beginning of Sect. 10.

Arguing that engineering functional decompositions are limited to only decompositions of functions into finite sets of basic functions (e.g. Lind 1994; Pahl and Beitz 1996; Stone and Wood 2000; Keuneke 1991) will also not do: the functions and functional compositions considered in the proof are rather simple.

Arguing against the proof by holding that engineers would typically reject the cases that are considered in the proof may also not work. One could argue that these cases are irrational from an engineering perspective: a transformation of a flow a to a flow d that is decomposed into transformations of a to b and then of b to a', as in Fig. 9, counts as an inefficient detour that typically is avoided in engineering; a transformation of a flow a to a flow b that is decomposed into itself and two transformations that cancel each other seems an even worse waste of resources. Yet, given that one of the goals of the formalisation of functions is the creation of computer tools for automated functional reasoning, these cases become again realistic. Such tools may generate cases in which functions are combined in a manner that with hindsight may be taken as irrational or inefficient. The case as given in Fig. 9 may be generated by a first step in which the functions $\phi_4 = \langle b, a' \rangle$, $\phi_5 = \langle a', d' \rangle$ and $\phi_6 = \langle d', c \rangle$ are combined to arrive at $\phi_2 = \mathrm{Comp}(\phi_4, \phi_5, \phi_6) = \langle b, c \rangle$, and by a second step in which the functions $\phi_1 = \langle a, b \rangle$ and $\phi_3 = \langle c, d \rangle$ are added. When seen in isolation, both steps seem reasonable explorations. And the case as given in Fig. 10 may similarly be generated by a first step in which the functions $\phi_1 = \langle a, b \rangle$ and $\phi_2 = \langle c, d \rangle$ are combined to arrive at $\mathrm{Comp}(\phi_1, \phi_2) = \langle \{a, c\}, \{b, d\} \rangle$, and by a second step in which the function $\phi_3 = \langle d, c \rangle$ is added, and both steps seem again reasonable explorations when taken in isolation. Hence, automated functional reasoning tools may generate the cases considered in the proof.

Nevertheless, there are ways of going around the proof, and one option is to label the composition relation for functions, say by the technical or physical principle X that is involved in the functional composition. Such a labelling makes engineering sense and has already been proposed by Kitamura and Mizoguchi (2003) since in their account, a *way of achievement* is added to each functional decomposition as was discussed in Sect. 2 (see Fig. 3). With such a labelling, the composition relation for token functions can be labelled as well, which in turn can be used to introduce parthood relations $\mathrm{P_X}(\phi, \phi')$ and $\mathrm{P_X}(\Phi, \Phi')$ that are conditional on the label. By now requiring that two part-of relations $\mathrm{P_X}(\phi, \phi')$ and $\mathrm{P_Y}(\phi'', \phi''')$, or $\mathrm{P_X}(\Phi, \Phi')$ and $\mathrm{P_Y}(\Phi'', \Phi''')$, may not be combined if they are defined relative to different composition principles $\mathrm{X} \neq \mathrm{Y}$, and by taking these principles X and Y sufficiently fine-grained, the cases uses in the proof may be defused. For instance, in the proof for type functions, Fig. 9, one may label the composition of the functions ϕ_4, ϕ_5 and ϕ_6 to ϕ_2 by a principle X, and the composition of the functions ϕ_1, ϕ_2 and ϕ_3 to ϕ by another principle Y. And one can do something similar for the proof for token functions, Fig. 10. It is another matter whether it indeed can be defended from an engineering point of view that part-of relations for technical functions defined relative to different composition principles may never be combined; yet, logically, such a requirement does provide a way to go around the proof. The result presented in this paper is then that the composition relations for technical functions and their subfunctions cannot be taken as defining unconditional formal parthood relations.

12 Conclusion and discussion

This paper aims at supporting the formalisation of functional descriptions as used in design methods. In Sect. 5, a modelling is given of technical functions of the two categories of token and type functions, and in Sect. 7, a modelling is given of the composition of token functions. In Sect. 10, a proof is presented that the relation between functions and their subfunctions cannot be taken as a formal parthood relation, that is to say, it cannot be unconditionally taken as such a parthood relation.

For design methodology in its current state, the consequences of these results are limited. Design methods advance different concepts of function, and designers may still use their preferred concept and maintain that subfunctions are parts of the functions they compose. The proof, when it applies to the concept of function used, shows then that this parthood relation should not be taken as a formal parthood relation but as an informal one for which, for instance, antisymmetry does not hold. Maintaining an informal parthood relation for functions may still be beneficial to designers, since it, as said in the introduction, suggests to designers to consider the design solutions to subfunctions as structural parts of the solutions to the overall functions. And that may be a helpful suggestion in designing.

There are, however, consequences for the development of design methodology. First, if design methodology is expected to develop to a state in which only one, single concept of function is advanced, then arguments are needed to rule out the other concepts that are currently in use in design methods, and the proof presented in this paper becomes relevant by providing such arguments. If one can argue, or simply decide, that the single concept of function to be adopted in design methodology should be such that the composition relation for functions is a formal parthood relation, then one can reject a number of the currently used concepts of function. Specifically, all concepts by which functions can be represented as transformations of flows of energy, material and signals are then ruled out.

Second, if the future state of design methodology is expected to be one in which functional descriptions are formalised, or one in which engineering ontologies or automated formalised reasoning tools standardly support functional descriptions, then all results presented in this paper become of use to the field. The modelling of token and type functions and the modelling of the composition of token functions provide formalisations of functional descriptions. And the proof shows that design methodology runs into contradiction when the relation between subfunctions and the functions they compose is unconditionally taken as a formal parthood relation.

Acknowledgments The modelling of functions builds on joint work with Stefano Borgo, Massimiliano Carrara and Pawel Garbacz. This work has been developed in part within the Marie Curie EuJoint project (IRSES 247503). Research for this paper was supported by The Netherlands Organisation for Scientific Research (NWO).

References

Arp R, Smith B (2008) Function, role, and disposition in basic formal ontology. Nature Precedings. http://precedings.nature.com/documents/1941/version/1. Accessed 24 October 2010

Borgo S, Carrara M, Garbacz P, Vermaas PE (2009) A formal ontological perspective on the behaviors and functions of technical artifacts. Artif Intell Eng Des Anal Manuf 23:3–21

Borgo S, Carrara M, Garbacz P, Vermaas PE (2011) A formalization of functions as operations on flows. J Comput Inf Sci Eng 11:031007

Bryant CR, McAdams DA, Stone RB, Kurtoglu T, Campbell MI (2006) A validation study of an automated concept generator design tool. Proceedings of ASME IDETC/CIE 2006, DETC2006-99489, Philadelphia

Burek P, Herre H, Loebe F (2009) Ontological analysis of functional decomposition. In: Fujita H, Mařík V (eds) New trends in software methodologies, tools and techniques: proceedings of the eighth SoMeT_09. IOS Press, Amsterdam, pp 428–439

Casati R, Varzi A (1999) Parts and places: the structures of spatial representation. MIT Press, Cambridge, MA

Chakrabarti A, Bligh TP (2001) A scheme for functional reasoning in conceptual design. Des Stud 22:493–517

Chandrasekaran B (2005) Representing function: relating functional representation and functional modeling research streams. Artif Intell Eng Des Anal Manuf 19:65–74

Chandrasekaran B, Josephson JR (2000) Function in device representation. Eng Comput 16:162–177

Erden MS, Komoto H, Van Beek TJ, D'Amelio V, Echavarria E, Tomiyama T (2008) A review of function modeling: approaches and applications. Artif Intell Eng Des Anal Manuf 22:147–169

Fantoni G, Apreda R, Bonaccorsi A (2009) Functional vector space. eProceedings of ICED 2009, Stanford, pp 2.311–2.322

Garbacz P (2006) A formal model of functional decomposition. Proceedings of ASME IDETC/CIE 2006, DETC2006-99097, Philadelphia

Garbacz P (2007) A first order theory of functional parthood. J Philos Log 36:309–337

Goel AK, Rugaber S, Vattam S (2009) Structure, behavior, and function of complex systems: the structure, behavior, and function modeling language. Artif Intell Eng Des Anal Manuf 23:23–35

Hirtz J, Stone RB, McAdams DA, Szykman S, Wood KL (2002) A functional basis for engineering design: reconciling and evolving previous efforts. Res Eng Des 13:65–82

Hubka V, Eder WE (1988) Theory of technical systems: a total concept theory for engineering design. Springer, Berlin

Johansson I (2004) On the transitivity of the parthood relations. In: Hochberg H, Mulligan K (eds) Relations and predicates. Ontos Verlag, Frankfurt, pp 161–181

Johansson I (2005) Formal mereology and ordinary language: reply to Varzi. Appl Ontol 1:157–161

Johansson I, Smith B, Munn K, Tsikolia N, Elsner K, Ernst D, Siebert D (2005) Functional anatomy: a taxonomic proposal. Acta Biotheor 53:153–166

Keet CM, Artale A (2008) Representing and reasoning over a taxonomy of part-whole relations. Appl Ontol 3:91–110

Keuneke AM (1991) Device representation: the significance of functional knowledge. IEEE Expert 6(2):22–25

Kitamura Y, Mizoguchi R (2003) Ontology-based description of functional design knowledge and its use in a functional way server. Expert Sys Appl 24:153–166

Kitamura Y, Kashiwase M, Fuse M, Mizoguchi R (2004) Deployment of an ontological framework of functional design knowledge. Adv Eng Inform 18:115–127

Kitamura Y, Koji Y, Mizoguchi R (2005) An ontological model of device function: industrial deployment and lessons learned. Appl Ontol 1:237–262

Lind M (1994) Modeling goals and functions of complex plants. Appl Artif Intell 8:259–283

Modarres M, Cheon SW (1999) Function-centered modeling of engineering systems using the Goal Tree–Success Tree technique and functional primitives. Reliab Eng Syst Saf 64:181–200

Ookubo M, Koji Y, Sasajima M, Kitamura Y, Mizoguchi R (2007) Towards interoperability between functional taxonomies using an ontology-based mapping. Proceedings of ICED 2007, paper no 154, Paris

Otto KN, Wood KL (2001) Product design: techniques in reverse engineering and new product development. Prentice Hall, Upper Saddle River, NJ

Pahl G, Beitz W (1996) Engineering design: a systematic approach. Springer, Berlin

Sasajima M, Kitamura Y, Ikeda M, Mizoguchi R (1996) A representation language for behavior and function: FBRL. Expert Syst Appl 10:471–479

Simons P (1987) Parts: a study in ontology. Clarendon, Oxford

Simons P, Dement C (1996) Aspects of the mereology of artifacts. In: Poli R, Simons P (eds) Formal ontology. Kluwer, Dordrecht, pp 255–276

Sridharan P, Campbell MI (2005) A study on the grammatical construction of function structures. Artif Intell Eng Des Anal Manuf 19:139–160

Stone R, Wood K (2000) Development of a functional basis for design. J Mech Des 122:359–370

Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T (1996) Supporting conceptual design based on the Function-Behaviour-State modeler. Artif Intell Eng Des Anal Manuf 10:275–288

Van Eck D (2010) On the conversion of functional models: bridging differences between functional taxonomies in the modeling of user actions. Res Eng Des 21:99–111

Varzi AC (2005) A note on the transitivity of parthood. Appl Ontol 1:141–146

Varzi A (2011) Mereology. In Zalta (ed) The Stanford encyclopedia of philosophy. http://plato.stanford.edu/entries/mereology/. Last accessed 27 September 2011

Vermaas PE (2009a) Parts, compositions and decompositions of functions in engineering ontologies". In: Ferrario R, Oltramari A (eds) Formal ontologies meet industry. IOS Press, Amsterdam, pp 34–45

Vermaas PE (2009b) The flexible meaning of function in engineering. eProceedings of ICED 2009 Stanford, pp. 2.113–2.124

Vermaas PE (2010) Subfunctions as parts of functions: some formal problems. Proceedings of ASME IDETC/CIE 2010, DETC2010-28095, Montreal

Vermaas PE, Garbacz P (2009) Functional decompositions and mereology in engineering. In: Meijers AWM (ed) Handbook philosophy of technology and engineering sciences. Elsevier, Amsterdam, pp 235–271

Vieu L (2005) On the transitivity of functional parthood. Appl Ontol 1:147–155

Vieu L, Aurnague M (2005) Part-of relations, functionality and dependence. In: Aurnague M, Hickmann M, Vieu L (eds) Categorization of spatial entities in language and cognition. John Benjamins, Amsterdam, pp 483–509

Winston M, Chaffin R, Herrmann D (1987) A taxonomy of part-whole relations. Cogn Sci 11:417–444