UNDERSTANDING AND REPRESENTING FUNCTIONS FOR CONCEPTUAL DESIGN

Yong Chen[1], Zhongqin Lin[1], Peien Feng[2], Youbai Xie[3]

[1]Institute of Automotive Engineering, School of Mechanical Engineering, Shanghai Jiao-Tong University, Shanghai, 200240, P. R. China
[2]State Key Laboratory of CAD&CG, School of Mechanical and Energy Engineering, Zhejiang University, Hangzhou, 310027, P. R. China
[3]Institute of Modern Design, School of Mechanical Engineering, Shanghai Jiao-Tong University, Shanghai, 200240, P. R. China

ABSTRACT

As a fundamental concept, function plays a crucial role in engineering design. Particularly, a formal functional representation methodology is indispensable for intelligent CAD (computer aided design) systems to reason about functions in a free way. This paper surveys various functional definitions systematically, resulting in the identification of the drawbacks of current understandings about function. A comprehensive functional definition is then given, based on which, functions are then classified as object-focused ones, process-focused ones and relation-focused ones. A formal functional representation methodology, composed of multiple approaches, is then presented to represent these kinds of functions. Cases demonstrate that the proposed approach can represent functions in an ontological way and is therefore convenient for computers to manipulate the functional information.

Keywords: Functional representation, conceptual design

1 INTRODUCTION

As the most crucial stage of the whole product development cycle, conceptual design is responsible for generating suitable principle solutions for desired functions and embodying them with some structural components [1]. Modern technical artefacts usually are very complex and contain multiple solution principles from different disciplines. Designers of Today, however, can only be taught knowledge belonging to single discipline or a few ones in the knowledge-explosion era. Therefore, it is often very difficult for a today's designer to fulfil the conceptual design of a modern product, which usually requires multi-disciplinary knowledge. Collaboration among multi-disciplinary designers could be a possible solution to the above issue. However, it is often unrealistic to call together the experts from all disciplines to give birth to an optimal conceptual design solution. To enhance the creativity of today's designers, it is then indispensable to provide them with an ideal computer aided conceptual design (CACD) system, which can search for optimum and innovative solution principles through the exploration in a wide solution space for desired functions without any bias.

Functions play a critical role during the design exploration process. To enable the exploration, it is indispensable to explicitly represent the desired functions in a computational manner so that computer can operate it freely without any ambiguities. Therefore, some efforts have been put into the functional representation research in recent years, with a result of some valuable fruits [2-8]. Note that functional representation discussed here is different from functional modelling, which usually aims at formalizing the functional achievement knowledge of a given device [9-11].

Existing functional representation approaches (for instance, verb-noun pair) are far from mature, which have prohibited computers from searching for correct and optimal principle solutions for desired functions. A major reason consists in that current understandings about the functional concept in the design society are limited but versatile. In addition, some commonsense knowledge that is difficult to be conceptualized also deteriorates the functional representation issue. Therefore, this paper is devoted to the development of a formal functional representation methodology.

The paper is organized as follows. After reviewing the existing understandings about the functional concept, section 2 clarifies what a function means for conceptual design and then classifies functions as object-focused functions, process-focused functions and relation-focused functions based on the new functional definition. The following three sections then elaborate how to represent these kinds of functions in a formal way, respectively. Section 6 then concludes this paper.

2 FUNCTIONAL CLARIFICATION

To develop a formal methodology for functional representation, it is necessary to explicitly state what a function is. As an overloaded concept, function, however, has been attributed to diverse definitions by different researchers in various areas. Before giving a uniform functional definition for intelligent CAD researches, this section, therefore, reviews the state-of-the-art of the understanding about the functional concept at first.

2.1 A survey of functional definitions

Among the diversified functional definitions, what are closely related to this paper are those from engineering design, artificial intelligence (AI) and the crossing area, intelligent CAD.

An important viewpoint in engineering design society argues that function represents a relation between the input and output of energy, material and information (Rodenacker, 1971) [14]. A major pitfall of this functional definition consists in that it emphasizes the objective behaviours of a system too much while ignoring its purposive aspects. Since a system often exhibits multiple behaviours simultaneously, it is then impossible for a designing agent to distinguish the desired function from its behaviours. For example, given a boiler in a thermal power station, it at least exhibits two primary behaviours, i.e. to heat water and to evaporate water. Without an explicit statement of purposive aspect, a designing agent would not be able to tell which behaviour was the desired function

However, function in AI community is often defined as "the relation between the goal of a human user and the behaviour of a system (Bobrow, 1984)" [17]. Contrary to the former definition, this definition underlines the purpose of a system too much while overlooking its objective properties. For instance, the function of a mechanical watch can be defined as to indicate time, rather than to generate a rotation with a uniform angular velocity. Functional definitions of such kind are not suitable for representing function in a formal way to support the computer-aided conceptual design of technical artefacts because the function of a technical device can vary when abstracted from different views. Instead of defining such kind of information as function, Rosenman and Gero have defined it as "purpose", a concept that merely exists in a social-cultural environment [19].

Engineering design is characterized as a function-driven process, where function bridges the gap between human purposes (customer needs) and objective behaviours of physical systems [1, 17, 19]. Therefore, function should treat both kinds of information equally, rather than emphasize one over the other. Thereby, the systematic approach to engineering design defines function as the general relationship between the input and output of a system aiming at performing a task [1]. Since the relationship between the input and output of a system is often called behaviour, this definition actually suggests representing functions with the intended part of all behaviours of a system via accentuating the aim of performing a task. Using such a functional definition, it is then easy to declare that the function of the boiler in a power station mentioned before is to evaporate water, which is further employed to drive the rotator.

To facilitate automated designing of physical devices, most intelligent CAD researches share similar functional definitions with the systematic approach, though employing different representation approaches [3, 12-13, 18, 20]. For example, Umeda et al fundamentally agreed with the above definition through defining function as "a description of behavior abstracted by human through recognition of the behavior in order to utilize the behavior" [12]. After making an explicit distinction between intrinsic and ascribed behaviours of a system, Goel and Stroulia further divided intrinsic behaviours into output behaviours that were directly observable and internal behaviours that caused those output ones and defined functions as those intended output ones [13]. Similarly, Sasajima et al employed "focus ports and objects" to discriminate functions from other system behaviours [3].

According to the above analysis, a conclusion can be made that design society today usually agrees to define function as the intended behaviour of a desired system.

2.2 A comprehensive functional definition

The existing cognition of function as intended behaviour, however, is too narrow and can't reflect its essence comprehensively. Since behaviour is usually represented as a transition of a flow or its state from input to output, the above understanding about function makes a designing agent merely focus on an output flow or its final state.

Designing focuses can be more than flows. Designing activities exist because the world around designing agents do not suit them, and the goal of designing agents is to change the world through the creation of artefacts [6, 19, 20], which implies that designing focus should be the world around designing agents. Therefore, we define **function as an intended transition of the world from a state sensed as unsatisfactory to a desirable one.**

Since the world is a combination of **objects** (i.e. flows in engineering design), **processes** that denote the changes of objects and the **relations** between objects, functions can then be classified as object-focused functions, process-focused functions and relation-focused functions with respect to different focuses. For example, the speed of a dissolving process could be sensed as too slow, a desired process-focused function could then be defined as to expedite the dissolving process. Another example is the relation-focused function, to separate dirt from clothes, whose primary focus is the change of the relation between dirt and clothes from input to output, instead of any requirement on the specific change of either material.

Since different kinds of functions have different focuses, the rest of the paper develops different formal approaches for their representations.

3 REPRESENTING OBJECT-FOCUSED FUNCTIONS

3.1 Overview

Object-focused functions mean that designing agents are concerned with the output objects or their output states. They have been the focus of functional representation and reasoning in the past engineering design researches.

Objects fall into the categories of material, energy and signal [1]. In the conceptual design of mechanical products, designing agents are primarily concerned with the transition of material and energy. Therefore, this paper merely deals with the formal representation approaches to functions related to these two kinds of objects.

Object-focused functions can be further classified as transformation-based functions and prohibition-based functions. A transformation-based function means that an input object is undesirable or is in an undesirable state and should be transformed into a desirable one, for example, to heat water. On the contrary, a prohibition function means that an input object is desirable or is in a desirable state while some potential EPEs (abbreviation of extensively physical effects, including physical effects, chemical effects, biologic effects, etc.), which would transform the input object into an undesirable state, should be prohibited. For example, the function, to support book, is to prohibit the fall of book due to the existentce of the gravity effect.

In what follows, how to represent the transformation-based functions and the prohibition-based functions are illustrated, respectively.

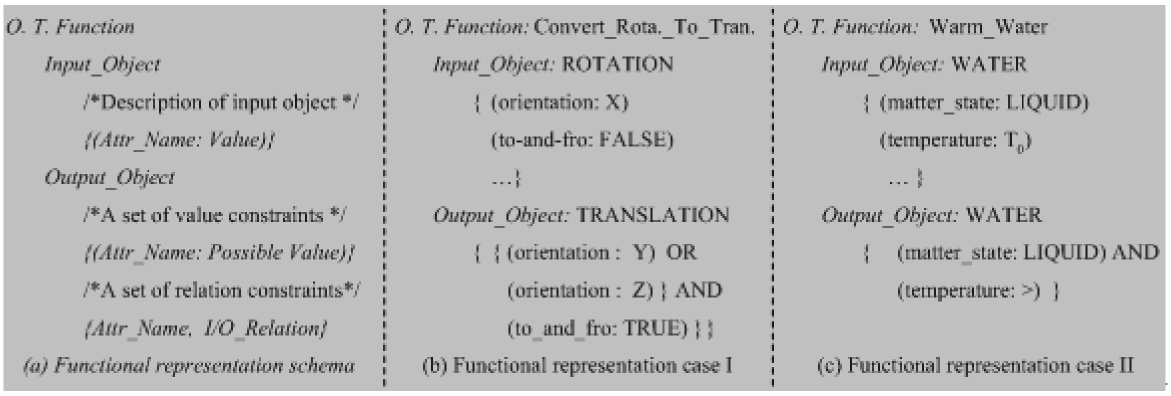
3.2 The representation of transformation-based functions

Traditionally, functions were usually represented with the verb-noun pairs in a semantic way. Although human designers could understand such representations without any difficulty, it is impossible for intelligent CAD systems to manipulate such functional information due to computers' poor ability to understand natural language.

According to the functional definition given by Rodenacker [15], intelligent CAD systems usually employ the input-output object pair, (O_{IN}, O_{OUT}), to represent functions formally to achieve automated conceptual design [21, 22]. For example, given a vector, *[Type, Orientation, To-and-fro]*, to denote a motional object (an energy object), a desired function can be represented as, (*[*Rotation*,* X*,* FALSE*]*, *[*Translation*,* Y*,* TRUE*]*), which can, for instance, be achieved by a slider-crank mechanism [22].

A major drawback of the above approach consists in that it requires the explicit representation of output objects. Since a conceptual design problem is often defined in an ill-structured way, it is often difficult for a designer to declare the output object or its state explicitly. It is the truth that there often exist a set of output states that could satisfy designer's demand on the output object. For example, a

designer maybe needs a to-and-fro translation in a horizontal plane from a given rotation. Since the horizontal plane means that the orientation of the desired translation can be in either *X*- or *Y*- direction, it is then impossible to represent the function with an input-output object pair.
Here a constraint-based approach is proposed to represent a function as a binary group, i.e. (O_{IN}, CO_{OUT}). Here O_{IN} represents the input object, while CO_{OUT} is a set of constraints on output objects. Constrains on the object attributes can be classified as two types: value constraints and relation constraints. A value constraint on an attribute prescribes the range of the value. For example, if a translation a translation in a horizontal plane is desired, then the following value constraint exists: ($orientation_{out}$ = “X”) OR ($orientation_{out}$ = “Y”). The value constraints are usually employed to represent the constraints on attributes with discrete values. A relation constraint on an attribute denotes a relation between the input and the output value of the corresponding attribute. For example, the function, to warm water, can be represented as the relation between the input and output temperature of water: $\text{temperature}_{out} > \text{temperature}_{in}$.
Based on the above cognition, a schema for representing object-transforming functions can then be developed, as shown in Figure 1. Here, object is represented with attribute-value, instead of a fixed vector model, so that different kinds of objects can be represented with different groups of attribute-values. Together with the schema, the representations of two functions are provided as well to illustrate the representation schema.



O. T. Function
Input_Object
/*Description of input object */
{(Attr_Name: Value)}
Output_Object
/*A set of value constraints */
{(Attr_Name: Possible Value)}
/*A set of relation constraints*/
{Attr_Name, I/O_Relation}
(a) Functional representation schema

O. T. Function: Convert_Rota._To_Tran.
Input_Object: ROTATION
{ (orientation: X)
(to-and-fro: FALSE)
…}
Output_Object: TRANSLATION
{ { (orientation : Y) OR
(orientation : Z) } AND
(to_and_fro: TRUE) } }
(b) Functional representation case I

O. T. Function: Warm_Water
Input_Object: WATER
{ (matter_state: LIQUID)
(temperature: T_0)
… }
Output_Object: WATER
{ (matter_state: LIQUID) AND
(temperature: >) }
(c) Functional representation case II

Figure 1. The representation schema of transformation-based functions

In a relational database, the above functional representation schema can be represented with the relational diagram shown in Figure 2.

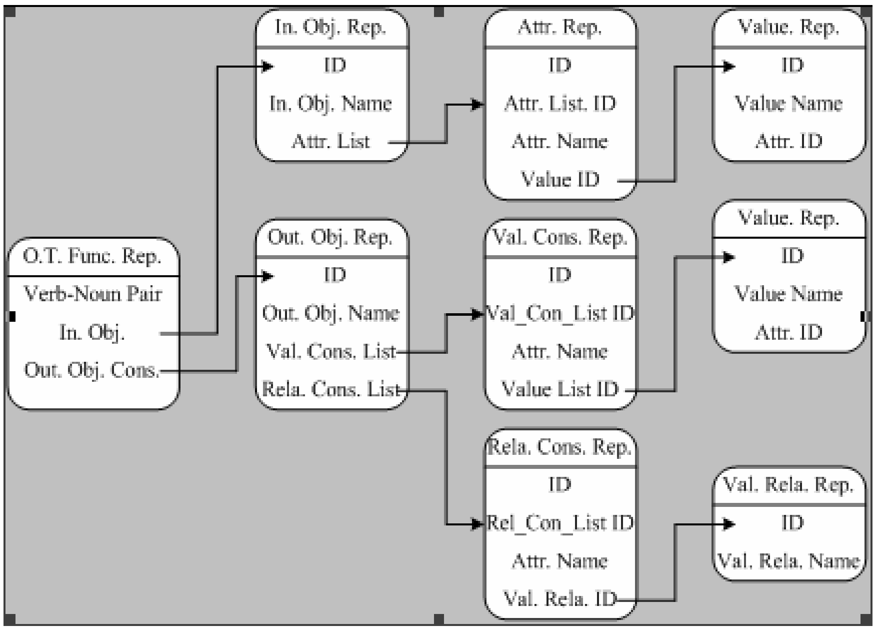
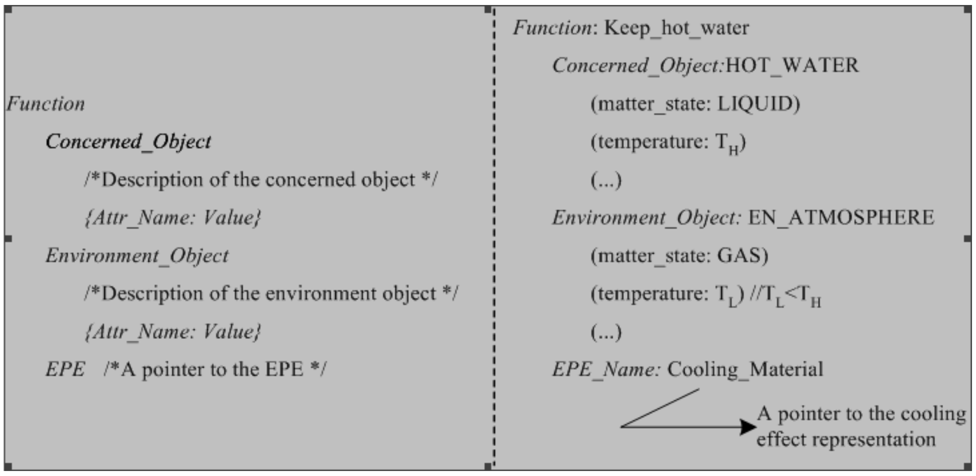


Figure 2. The relational diagram for representing transformation-based functions

Compared with the traditional input-output object pair, the proposed functional representation approach doesn't require a comprehensive representation of output objects and therefore are more flexible.

3.3 The representation of prohibition-based functions

As stated before, a prohibition-based function is to prevent a potential EPE from changing an object into an undesirable state. Therefore, a triple, <*O*, *E*, E_P>, can be employed to represent a prohibition-based function, where *O* represents the concerned object, *E* represents the environment object that interacts with the concerned object, while E_P denotes the EPE that should be prohibited. Figure 3 shows the representation schema of prohibition-based functions and a representation case.



```
Function
    Concerned_Object
        /*Description of the concerned object */
        {Attr_Name: Value}
    Environment_Object
        /*Description of the environment object */
        {Attr_Name: Value}
    EPE   /*A pointer to the EPE */
```

```
Function: Keep_hot_water
    Concerned_Object:HOT_WATER
        (matter_state: LIQUID)
        (temperature: T_H)
        (...)
    Environment_Object: EN_ATMOSPHERE
        (matter_state: GAS)
        (temperature: T_L) //T_L<T_H
        (...)
    EPE_Name: Cooling_Material
                A pointer to the cooling
                effect representation
```

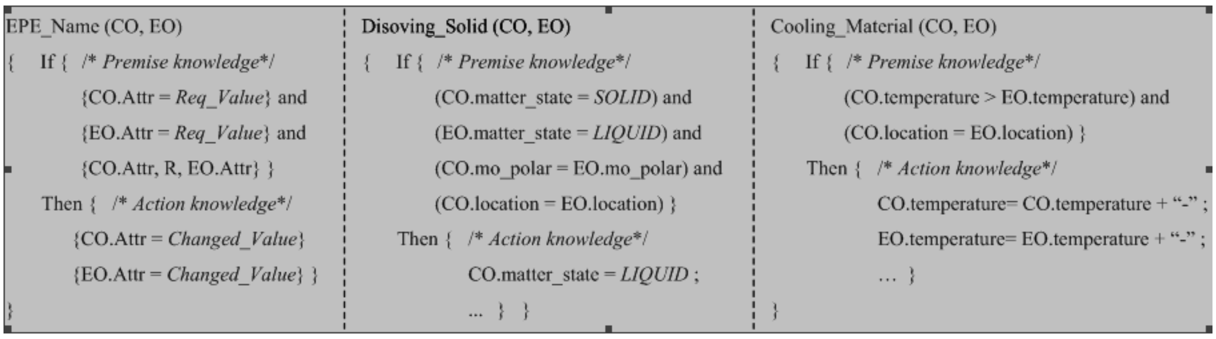
Figure 3. The representation schema of prohibition-based functions

Since object is represented with a set of attribute-value pairs, the primary task here is then to represent an EPE. An EPE can be regarded as a kind of interaction between an object and its environment. For example, the EPE, *cooling_material,* represents a heat-exchanging interaction between an object with higher temperature and its environment with lower temperature. Therefore, an EPE can be represented with two kinds of knowledge, i.e. premise and action.

The premise knowledge denotes under what conditions the corresponding EPE can take place. The premise knowledge falls into three categories: the attribute constraints on the concerned object, the attribute constraints on the environment object, the constraints on the relation between the attribute of the concerned object and that of the environment object. Given the solid-dissolving effect as an example, a constraint on the concerned object is that its attribute, *matter_state*, should be with the value of "SOLID"; a constraint on the environment object is that its matter-state should be with the value of "LIQUID"; the attribute relation constraints include that the molecular polarity attribute of both objects should be equal and that the values of their place attributes should be identical.

The action knowledge denotes how the interaction of an EPE will change the concerned object and the corresponding environment object. When the attribute of the concerned object that is changed is an enumeration variable, the action knowledge can be directly represented as the final attribute value. For example, the action knowledge of the dissolving effect mentioned before can be represented as: matter-state = LIQUID, which means that the final matter-state of the concerned object is "LIQUID". However, when the attribute is a continuous variable, it is then impossible to represent the action knowledge with the above approach. For example, since the temperature attribute of an object can't be represented in an enumeration manner, the action knowledge of the heating effect then can't be represented with a prescribed value. Since the value change of a continuous attribute is either increase or decrease, it can then be represented in a qualitative manner through adding the symbol, + or -, to the original value. For example, the decrease of the temperature of a material can be represented as, T_0-, where T_0 represents the original temperature of the material.

Since an EPE is an abstraction of the general interaction between a concerned object and its environmental object, it can be represented as a procedure, which takes the initial objects as parameters and changes their states according to the action rules. Figure 4 shows the EPE representation schema, together with the effects of dissolving solid and cooling material as two examples.



```
EPE_Name (CO, EO)
{   If {  /* Premise knowledge*/
          {CO.Attr = Req_Value} and
          {EO.Attr = Req_Value} and
          {CO.Attr, R, EO.Attr} }
    Then {  /* Action knowledge*/
          {CO.Attr = Changed_Value}
          {EO.Attr = Changed_Value} }
}
```

```
Disoving_Solid (CO, EO)
{   If {  /* Premise knowledge*/
          (CO.matter_state = SOLID) and
          (EO.matter_state = LIQUID) and
          (CO.mo_polar = EO.mo_polar) and
          (CO.location = EO.location) }
    Then {  /* Action knowledge*/
          CO.matter_state = LIQUID ;
          ... }  }
```

```
Cooling_Material (CO, EO)
{   If {  /* Premise knowledge*/
          (CO.temperature > EO.temperature) and
          (CO.location = EO.location) }
    Then {  /* Action knowledge*/
          CO.temperature= CO.temperature + "-" ;
          EO.temperature= EO.temperature + "-" ;
          ... }
}
```

Figure 4. The formal representation schema of EPE

Therefore, the prohibition-based functions should be represented as two separate parts in the intelligent CAD system. The functional schema can be represented with aid of the relational database, while the EPE should be represented as a procedure base. As a result, the diagram for representing prohibition-based functions can be shown in Figure 5. Since EPE are represented in a generalized manner and are independent of any specific function, EPE can be inherited from one function to another. Therefore, the EPE is collected in the procedure base.

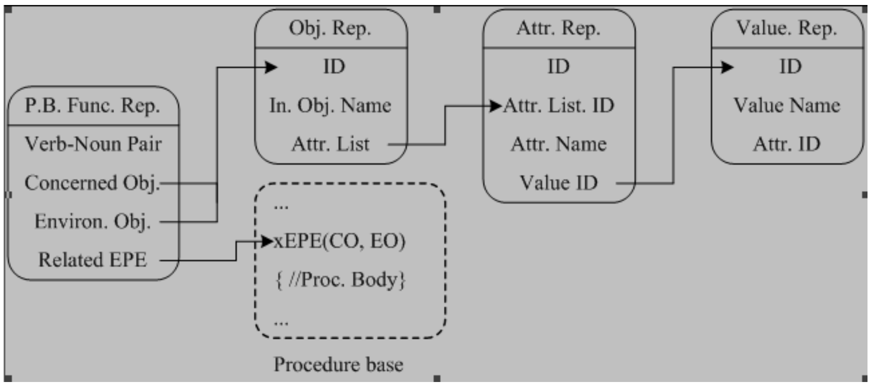


Figure 5. The representation diagram of prohibition-based functions

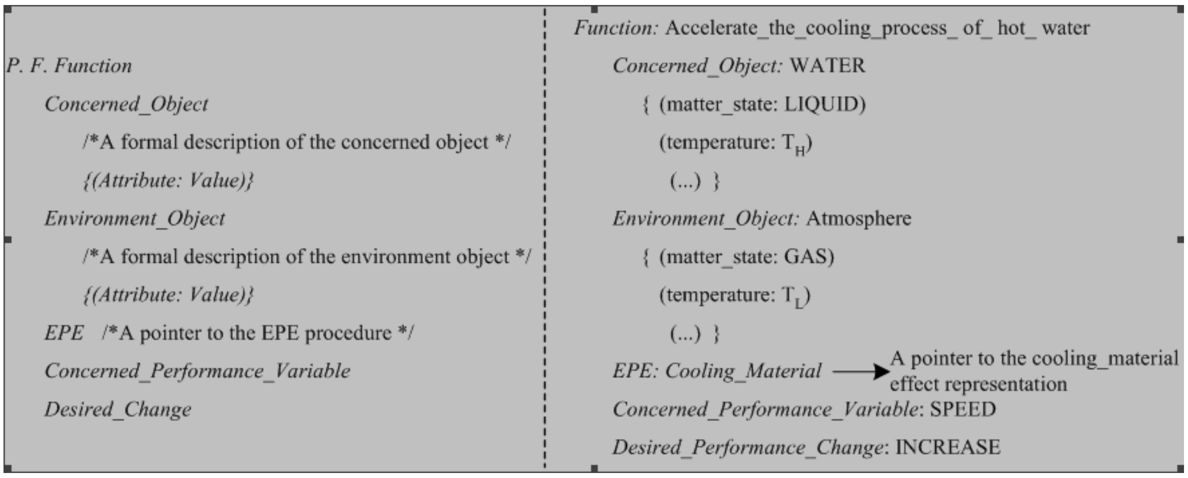
With aid of the above representation approaches, intelligent CAD systems can then manipulate the information of the prohibition-based functions correctly. Compared with the traditional approach based on the verb-noun pairs, the proposed approach can represent functions more accurately. Even some similar functions can be clearly distinguished. For example, two semantically similar functions, to store water (i.e., to prevent water from flowing away), and the function, to preserve water (i.e., to prevent water from decay), can be distinguished from each other when represented with the proposed approach.

4 REPRESENTING PROCESS-FOCUSED FUNCTIONS

The process-focused functions originate from dissatisfactions with the performances of interested processes, which is different from object-focused functions resulting from the dissatisfactions with the object states.

To represent process-focused functions, it is indispensable to collect possible performance variables of processes. Although there are various processes, the performance variables are very few. The often-used variables include the transformation quantity, the transformation speed and the transformation ratio. Here, the transformation quantity is equivalent to the quantity of output flows, while the transformation ratio equals to the ratio of the quantity of the output flow to that of the input flow. For example, designers can be dissatisfied with that the hot water cools too slowly, which means that a process-focused function, to accelerate the cooling speed, is desired. Another example is that a designer is dissatisfied with that the dissolubility of a material in water is too small, and therefore a process-focused function, to improve the dissolution ratio of the material, is desired.

As there are merely two possibilities to change a process, e.g. to improve and to restrain, a performance variable can then be with the value "INCREASE" or "DECREASE".
A process-focused function should be represented with a group composed of five kinds of knowledge, $<O_C, O_E, E_P, V_P, C>$. Here, O denotes the concerned object, E_O represents environment object, E_P is the EPE that results in the process of interest, V_P is the concerned performance variable and C is the desired change of the performance variable. A schema for representing process-focused functions is shown in Figure 6.



P. F. Function
  Concerned_Object
    /*A formal description of the concerned object */
    {(Attribute: Value)}
  Environment_Object
    /*A formal description of the environment object */
    {(Attribute: Value)}
  EPE /*A pointer to the EPE procedure */
  Concerned_Performance_Variable
  Desired_Change

Function: Accelerate_the_cooling_process_ of_ hot_ water
  Concerned_Object: WATER
    { (matter_state: LIQUID)
    (temperature: T_H)
    (...) }
  Environment_Object: Atmosphere
    { (matter_state: GAS)
    (temperature: T_L)
    (...) }
  EPE: Cooling_Material $\longrightarrow$ A pointer to the cooling_material effect representation
  Concerned_Performance_Variable: SPEED
  Desired_Performance_Change: INCREASE

Figure 6. The representation schema of process-focused functions

An example is also given in Figure 6 to illustrate the representation model. In this example, what is concerned is that the cooling speed of hot water is too slow. The functional requirement, to expedite the cooling process is represented through setting the performance variable, *SPEED*, as *INCREASE*.
In the relational database system, the above schema can be represented with the diagram shown in Figure 6. Using this model, it is then possible to represent process-focused functions explicitly, which will then facilitate intelligent CAD systems to manipulate functional information accurately.

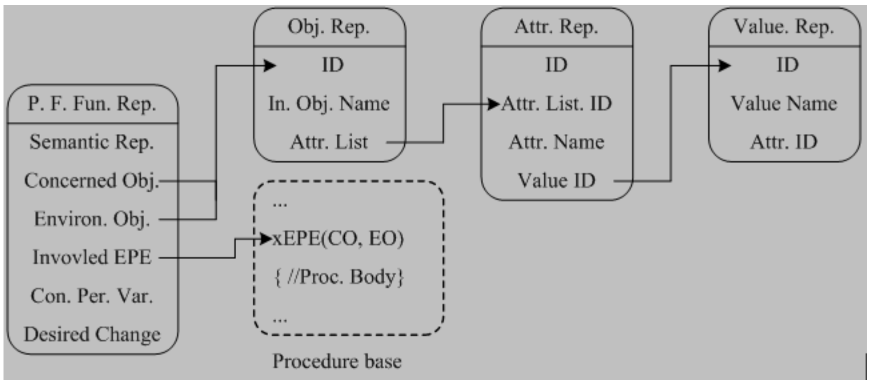


Figure 7. The representation diagram of process-focused functions

Based on the functional representation approach, some process-focused functions can then be differentiated from those object-focused functions, which, however, could be represented with similar phrases when described in an informal way. For example, the function, to store food, can be understood either as to prevent food from decaying, which is an object-focused function, or as to slow down the decay speed of food, which is a process-focused function.

5 REPRESENTING RELATION-FOCUSED FUNCTIONS

There are various kinds of relations between the objects in the world, for example, the orientation relation (parallel, perpendicular, etc), the magnitude relation (=, >, <, etc), the location relation (same, different), the order relation, the group relation, etc. Correspondingly, there are many relation-focused functions that deal with the changes of relations between multiple objects from input to output. To represent relation-focused functions, it is desirable to represent relations between objects at first.

5.1 Representation of relations

The relations between two objects can be represented in either a qualitative or a quantitative manner. Formulae are usually employed to represent the quantitative relations. Since conceptual design usually deals with qualitative change of relations, only qualitative relations are taken into consideration here. The qualitative relations between objects can be represented as abstract mathematical symbols, such as "=", "≠", ">", "<", etc.

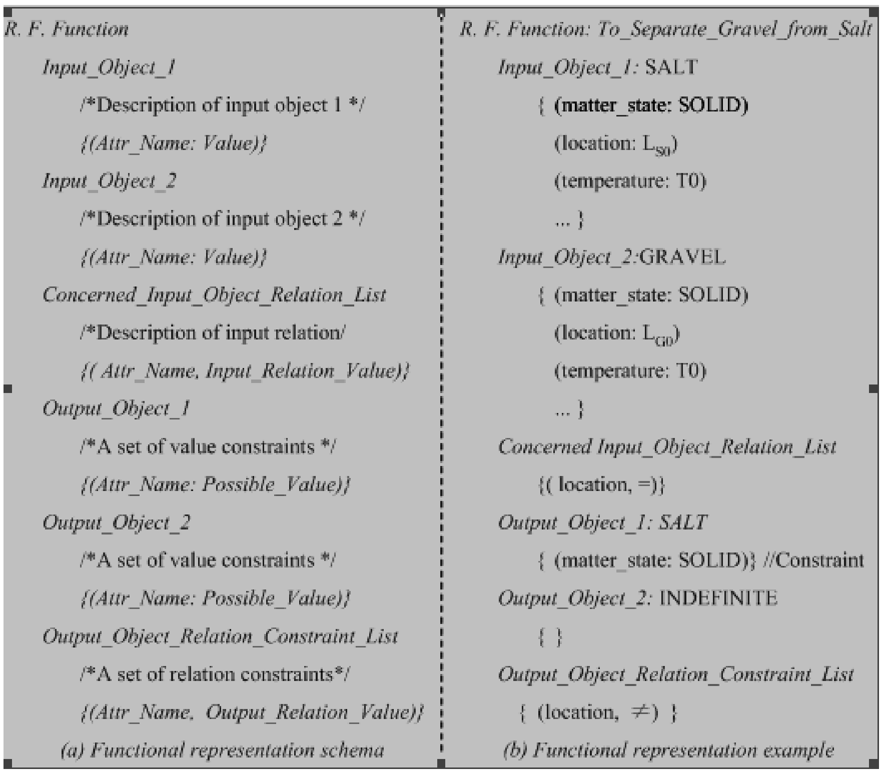
A relation in itself should depend on a concrete attribute of an object. For example, when two objects are regarded as to be in a mixed relation, a designing agent actually thinks that they are in the same location, where the related attribute is "location".

Therefore, a relation between two concerned objects can be represented as a constraint on a specific attribute between them, i.e. $<O_1, O_2, A, V>$. Here O_1 and O_2 represent the related objects, A is the attribute that the relation depends and V is the relation value. For example, the mixed relation between salt and gravel can be represented as <salt, gravel, location, =>, which means that salt and gravel are in the same location.

5.2 Representation of relation-focused functions

Based on the above relation representation, it is then possible to represent relation-focused functions in a computational way. Theoretically speaking, a relation-focused function can be represented in a binary group, $<R_{in}, R_{out}>$, where R_{in} represents the relation between input objects and R_{out} represents the relation between output objects. However, if the above relation representation approach were employed to represent R_{out} in a forma way, it would imply that the output objects should be explicitly represented as well. In a practical situation, however, it is often impossible to define an output object explicitly. For example, when a designing agent want to define the function, to separate gravel from salt, in a formal way, it is very possible that he doesn't care what state the gravel is in. Hereby, what he really cares is that the gravel (the impurity) should be separated from salt. Therefore, the constraint-based method similar to that employed in the representation of transformation-based functions is also employed here to denote the output objects.

The representation schema of relation-focused functions is shown in Figure 8, together with the function, to separate gravel from salt, as an example. How to store the functional information in a relational database is similar to that shown in Figure 2. Due to the limited space, it will not be elaborated here.



R. F. Function
Input_Object_1
/*Description of input object 1 */
{(Attr_Name: Value)}
Input_Object_2
/*Description of input object 2 */
{(Attr_Name: Value)}
Concerned_Input_Object_Relation_List
/*Description of input relation/
{(Attr_Name, Input_Relation_Value)}
Output_Object_1
/*A set of value constraints */
{(Attr_Name: Possible_Value)}
Output_Object_2
/*A set of value constraints */
{(Attr_Name: Possible_Value)}
Output_Object_Relation_Constraint_List
/*A set of relation constraints*/
{(Attr_Name, Output_Relation_Value)}
(a) Functional representation schema

R. F. Function: To_Separate_Gravel_from_Salt
Input_Object_1: SALT
{ (matter_state: SOLID)
(location: L_{S0})
(temperature: T0)
... }
*Input_Object_2:*GRAVEL
{ (matter_state: SOLID)
(location: L_{G0})
(temperature: T0)
... }
Concerned Input_Object_Relation_List
{(location, =)}
Output_Object_1: SALT
{ (matter_state: SOLID)} //Constraint
Output_Object_2: INDEFINITE
{ }
Output_Object_Relation_Constraint_List
{ (location, ≠) }
(b) Functional representation example

Figure 9. The schema for representing relation-focused functions

Compared with the traditional approach for representing relation-focused functions with verb-noun pairs, this approach can provide a computational foundation for intelligent CAD systems to smoothly manipulate relation information formalized as attribute relations between two concerned objects. For example, when the above function, to separate gravel from salt, is formalized as the change of the location relation from "=" to "≠", intelligent designing agents can then focus on the corresponding attribute relation change.

6 CONCLUSIONS

This paper summarized the state of the art of functional understanding at first, and then proposed a comprehensive function definition. Based on the proposed definition, functions are then classified as object-focused functions, process-focused functions and relation-focused functions. Multiple formal approaches, which comprise a comprehensive functional representation methodology, are developed for the representation of these kinds of functions, respectively. Functional representation cases demonstrate that the proposed methodology can explicitly represent functions by means of relating specific attributes of interest not only to object-focused functions, but also to process-focused functions and relation-focused functions, which, however, could merely be represented with traditional approaches in an informal way (such as verb-noun pairs with or without modifiers). Therefore, it provides a computational foundation for intelligent CAD systems to manipulate functional information freely, which will not only facilitate the exact retrieval of solution concepts from design knowledge base for routine design, but also will bridge the gap between function and behaviour for creative design. The methodology proposed in this paper can actually be regarded as an ontological approach to represent functions. To develop a functional representation ontology, our future work will collect a set of terms (such as attributes and their values for representing physical objects), and will also continuously develop computational design process models to support such activities.

ACKNOWLEDGEMENTS

The research work introduced in this paper is supported by Natural Science Foundation of China under the grant No. 50505025 and 50335040. The authors also gratefully acknowledge the anonymous reviewers who have given valuable suggestions to improve the quality of this paper.

REFERENCES

[1] Pahl G. and Beitz W. *Engineering Design: A Systematic Approach* (2nd Edition), 1996 (Springer, Berlin).
[2] Chakrabarti A. and Bligh T. An approach to functional synthesis of solutions in mechanical conceptual design. Part I: knowledge representation. *Research in Engineering Design*, 1994, 6(3), 127-141.
[3] Sasajima M., Kitamura Y., Ikeda M. and Mizoguchi R. A representation language for behavior and function: FBRL. *Expert Systems with Applications*, 1995, 10(3-4), 471-479.
[4] Kirschman C. and Fadel G. Classifying functions for mechanical design. *Journal of Mechanical Design*, 1998, 120, 475-482.
[5] Szykman S., Racz J., Bocheneck C. and Sriram R. D. A web-based system for design artefact modelling. *Design Studies*, 2000, 21, 145-165.
[6] Chandrasekaran B. and Josephson J. R. Function in device representation. *Engineering with Computers*, 2000, 16(1), 162-177.
[7] Hirtz J., Stone R. B., McAdams D. A., Szykman S. and Wood K. L. A functional basis for engineering design: Reconciling and evolving previous efforts. *Research in Engineering Design*, 2002, 13, 65-82.
[8] Deng Y.-M. Function and behaviour representation in conceptual mechanical design. *AI EDAM*, 2002, 16(5), 343-362.
[9] Keuneke A. M. Device representation: The significance of functional knowledge. *IEEE Expert*, 1991, 6(2), 22-25.
[10] Chandrasekaran B., Goel A. K. and Iwasaki Y. Functional representation as design rationale. *IEEE Computer*, 1993, 7(1), 48-56.
[11] Iwasaki, Y., Vescovi, M., Fikes, R., Chandrasekaran, B., 1995, Causal Functional Representation Language with Behavior-based Semantics. Applied Artificial Intelligence, 9: 5-31.

[12] Umeda Y., Ishii M., Yoshioka M., Shimomura Y. and Tomiyama T. Supporting conceptual design based on the function-behaviour-state modeller, *AI EDAM*, 1996, 10, 275-288.
[13] Goel A. K. and Stroulia E. Functional device models and model-based diagnosis in adaptive design, *AI EDAM*, 1996, 10, 355-370.
[14] Rodenacker W. *Methodisches Konstruieren*, 1991(Springer, Berlin).
[15] Miles L. D. *Techniques of Value Analysis and Engineering*, 1972(McGraw Hill, New York).
[16] Bobrow D.G. Qualitative reasoning about physical systems: An introduction. *Artificial Intelligence*, 1986, 24, 1-5.
[17] Suh N. P. Axiomatic design: Advances and applications, 2000(Oxford University Press).
[18] Kitamura Y., Sano T., Namba K. and Mizoguchi R. A functional concept ontology and its application to automatic identification of functional structures. *Advanced Engineering Informatics*, 2002, 16, 145-163.
[19] Rosenman M. A., Gero J. S., 1998. Purpose and function in design: from the social-cultural to the techno-physical. Design Studies, 19, 161-186.
[20] Gero J. S. Design prototypes: A knowledge representation schema for design. AI Magazine, 19990, 11(4), 26-36.
[21] Welch, R. V., Dixon, J. R., 1994. Guiding Conceptual Design through Behavioral Reasoning, Research in Engineering Design, 6(3) : 169-188.
[22] Chen Y., Feng P., He B., Lin Z. and Xie Y. Automated conceptual design of mechanisms using improved morphological matrix. *ASME Journal of Mechanical Design*, 2006, 128(3), 516-526.
[23] Feng, P.-E., Xu, G.-R., Zhang, M.-J., 1996, Feature Modeling based on Design Catalogues for Principle Conceptual Design, AI EDAM, 10(4): 347-354.

Contact: Y. Chen
Institution/university: Shanghai Jiao-Tong University
Department: Institute of Automotive Engineering, School of Mechanical Engineering
Street: 800#, Dongchuan road
City: Shanghai
Country: P. R. China
Phone: +86-21-3420-6786
Fax: +86-21-3420-4542
E-mail: mechenyong@sjtu.edu.cn