A scheme for functional reasoning in conceptual design

Amaresh Chakrabarti and *Thomas P. Bligh*, The Engineering Design Centre, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

An ideal functional reasoning environment should support designs of any nature, routine or innovative, at any level of detail, as well as through varying levels of detail. In this paper, three existing functional reasoning models are reviewed in this perspective. It has been found that none of these models support all of these requirements. It has been shown that a functional reasoning approach cannot guarantee the generation of solution concepts, which are combinations of known solutions, unless guided by the knowledge of existing solutions. A new model which can support design both across a level of detail and down through levels of detail has been proposed, which, using a divide and rule approach and using recursive problem redefinition while incorporating existing solutions, could support conceptual design. It is also shown that, although the generation of completely new solutions is not supported by the model, the model, when aided by a framework allowing a sustained progress of its knowledge base by transfer of knowledge from existing designs in the form of basic structures and rules of combination, could support generation of designs which otherwise would be considered unsupportable in a systematic way (innovative). *© 2001 Elsevier Science Ltd. All rights reserved.*

Keywords: functional reasoning, conceptual design, engineering design, innovative design

The central objective of engineering design is to generate physical descriptions of designs, sufficient for their implementation, which would provide the intended functions of the problem. Therefore, it is the knowledge of the intended functions that should constrain the design process which leads to these physical descriptions. In other words, designing should be guided by functions. Investigation into how the knowledge of functionality (of design problems and possible solutions) can be used to guide, constrain and shape the design activity, is an important strand of present design research.

www.elsevier.com/locate/destud
0142-694X/01 $ - see front matter *Design Studies* **22** (2001) 493–517
PII: S0142-694X(01)00008-4
© 2001 Elsevier Science Ltd All rights reserved Printed in Great Britain

One goal of this research is to understand the conceptual design activity (i.e. how functional requirements of a design problem are transformed into schematic descriptions of design solution concepts), and to develop computational methods to support this synthesis process. This presupposes a formal model of the conceptual design process. Any formal model of the conceptual design process, therefore, requires a *common language* in which design problems and solutions can both be described, compared and modified. *Functional reasoning approaches*[1–6] are promising for providing one such common language, in terms of a functional description of the problems and solutions. The purpose of this paper is to examine three major existing functional reasoning models, to identify their shortcomings, and to propose an alternative scheme to eliminate these problems. The approach is prescriptive in nature, and should help primarily towards better computational support to conceptual design. The work does not claim to describe, model or explain conceptual design practised by designers, although some overall similarity can be observed (see Section 3).

1 *Three existing functional reasoning approaches*

The word *function* is regarded here as a description of the action or effect required by a design problem, or that supplied by a solution. A *functional representation*, therefore, should allow one to describe design problems and solutions in terms of their functions. For example, the intended function of a design problem could be that of transmitting torque; one solution to this problem would be to use a shaft. The idea of functional reasoning in conceptual design is to reason at the functional level in order to generate solutions to specified design problems, and to evaluate these. Each model for functional reasoning consists of two parts:

- a functional representation of the objects to be reasoned about, and
- a reasoning scheme.

There are two existing functional representations. One is a natural-language-like representation, where verbs are used to describe what a strucutre does, or is supposed to do[1,7,8]. An example would be this description: a *shaft* (strucutre) *transmits torque* (function). An advantage of this representation is that it is close to the way in which designers express their ideas. However, in general, natural language lacks precision. It is difficult to formalise this representation in a generalised way, and there are many compound functions for which a standard name such as *transmit* does not exist.

The other representation is a mathematical representation of function, where it is expressed as a transformation between input and output. It is formalisable, and therefore is more suitable for a computational environ-

1 **Freeman, P and Newell, A** 'A model for functional reasoning in design', in *Proceedings of the Second International Joint Conference on Artifical Intelligence, London* (1971) pp 621–633

2 **Yoshikawa, H** 'General design theory and a CAD system', in **T Sata and E Warman** (eds) *IFIP Man–machine communication in CAD/CAM* (1981) pp 35–88

3 **Yoshikawa, H** 'Design theory for CAD/CAM integration' *Annals of the CIRP* Vol 34 No 1 (1985) 173–178

4 **Grabowski, H and Benz, T** 'Functional modelling in intelligent CAD systems', in *Preprints of the Second IFIP WG 5.2 Workshop on Intelligent CAD, Cambridge, England* (1988) pp 179–196

5 **Grabowski, H and Benz, T** 'Implementing the design methodology', in *Preprints of the Third IFIP WG 5.2 Workshop on Intelligent CAD, Osaka, Japan* (1989) pp 17–37

6 **Schmekel, H** 'Functional models and design solutions' *Annals of the CIRP* Vol 38 No 1 (1989) 129–132

7 **Johnson, A L** 'Functional modelling: a new development in computer-aided design', in *Proceedings of the IFIP WG 5.2 Workshop on Intelligent CAD, Cambridge, UK*, North-Holland, Amsterdam (1988)

8 **Lai, K and Wilson, W R D** 'FDL—a language for function description and rationalization in mechanical design' *ASME Journal of Mechanisms, Transmissions and Automation in Design* Vol 111 (1989) 117–123

ment. However, if a man–machine environment is to be provided using this representation, the commonly used functions expressed in the first representation would have to be mapped into the latter representation before any general functional reasoning support environment could be developed. An example is shown in Figure 1 in which the function *transmit* is shown in these two representations. From here on, the word 'function' will be used to describe any function covered by the above representations, with the assumption that the definition of each such function is precisely known.

Historically, there have been three influential approaches to functional reasoning in design. The earliest one[1] suggests that each given *structure* (a structure, which could be a physical entity, that has attributes which enable it to provide specified functions) be described by the functions it can *provide*, and to provide each of these, the functions that it *requires* (Figure 2a). The search for a solution to a given problem, defined in terms of a given set of desired functions, would start with a search among known structures (as outlined above) for the ones that can provide the desired functions. These chosen structures, in turn, require some other functions in order that they can provide the desired functions. Now, new structures would be searched for, so as to provide these required functions, which in turn give rise to new functional requirements. This process (see Figure 2b) would continue until all the functions required are provided by some structures. Each resulting combination of structures evolved by the above process thereby becomes a solution to the design problem posed at the beginning.

The second model is the *paradigm model* by Yoshikawa[2,3]. In this model, a design problem is expressed in terms of a set of functional requirements (such as the design of an animal which *runs* fast, *peeps*, and *swims* fast), and the solutions in terms of a set of attributes which enable specific functions to be met (such as a modified dog having frog's legs as front legs, and a bird's beak instead of a dog's mouth). The designer proposes a solution which seems to fulfil the requirements best, say a dog which at least runs fast. He then evaluates it by identifying the *wrong components* of the solution which make it not satisfactory for the specification, and formulates the specification still to be met from the difference between the

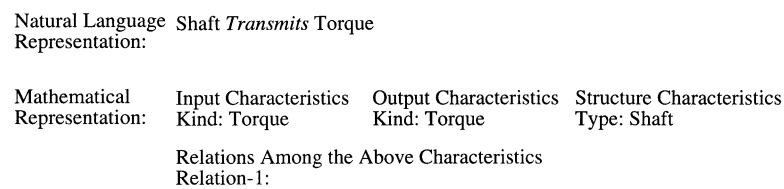


Figure 1 Two equivalent representations of the same function

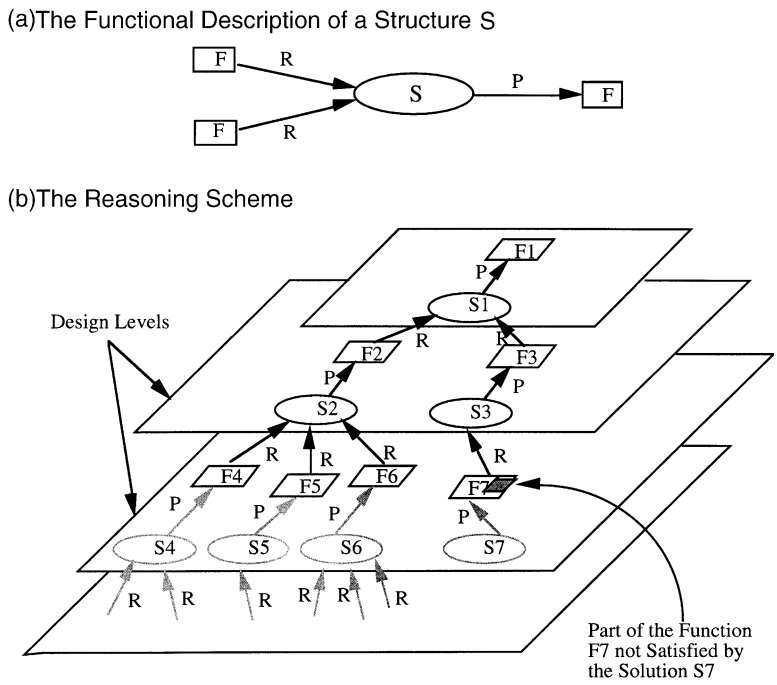


Figure 2 A pictorial representation of the functional reasoning model by Freeman and Newell

F1, F2 : Functions Provided or Required by a Structure
S1, S2 : Structures
P: Indicates to the Functions Provided
R: Indicates from the Functions Required

original specification and the function of the provisional solution. He then searches for new provisional components which would reduce the difference, and replaces the wrong components with the new ones, i.e. beak, to form a second and better provisional solution. This process is continued, such as the front feet being replaced with webbed feet, until the solution is satisfactory.

In the widely accepted model by Pahl and Beitz[9], which henceforth will be referred to as the *systematic model*, a design problem should be expressed as a (set of) *solution-neutral* function(s). These functions are then progressively expanded into combinations of sub-functions. These combinations are called *function-structures*. This process of *simplification* is continued until the *sub-functions* (functions of which a function-structure is composed) of these function-structures are *sufficiently simple*. Further variants to these function-structures could be generated by recombinations of their constituent sub-functions. The optimum function-structure is now chosen, and the possible *solution-alternatives* (i.e. structures, in the sense of the previous model) to each sub-function in the optimum function-struc-

9 Chakrabarti, A and Bligh, T P 'A scheme for functional reasoning in conceptual design', in *CUED technical report CUED/C-EDC/TR-60* Cambridge University Engineering Department, Cambridge (1998)

ture is found. These solutions are combined into alternative *solution concepts*, which are then evaluated, and the most promising ones chosen. A schematic of the process is shown in Figure 3.

2 *A critique of the three functional reasoning approaches*

An ideal functional reasoning approach should cater for at least three distinct requirements. Firstly, it should support designs of any nature, ranging from *routine* to *innovative*. Secondly, it should support the synthesis of solution concepts to a given design problem at a given level of design.

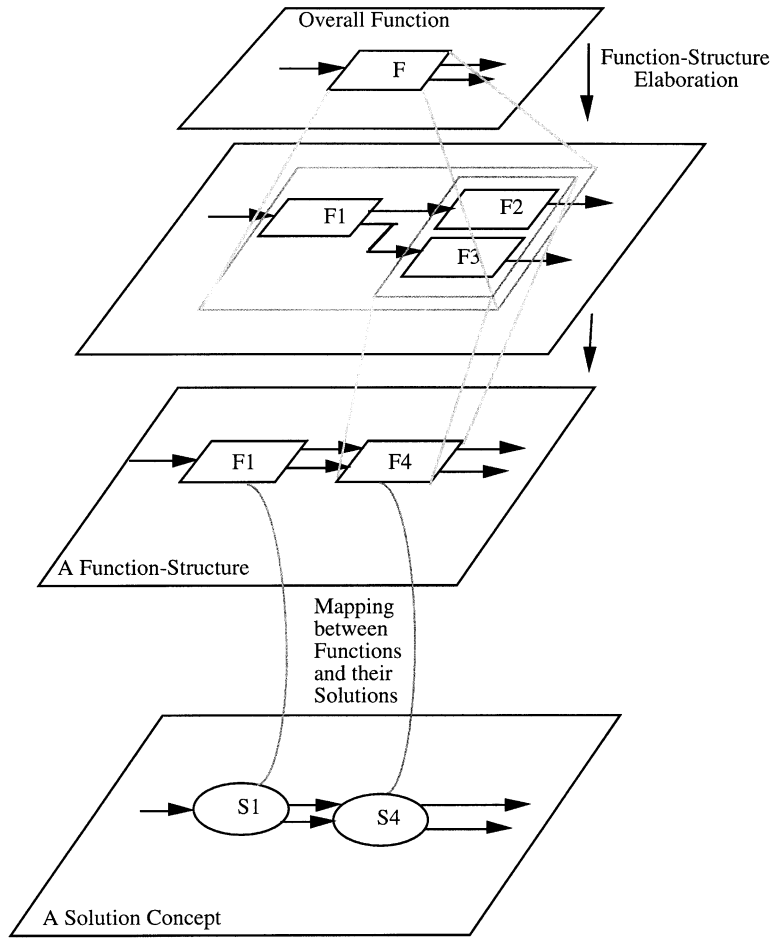


Figure 3 A pictorial representation of the systematic model for functional reasoning

F1, F2... : Sub-Functions Constituting a Function-Structure
S1, S2... : Sub-Solutions Constituting a Solution-Concept

Finally, it should support the elaboration of the solution concepts through subsequent levels of detail.

2.1 Freeman and Newells' model

Freeman and Newell's model is essentially a process of how a design can be evolved down through levels of detail. However, this process does not help designing at a given level, when, as is often the case, no structure can be found which will completely provide the desired functions; see Fig. 2 where a part of the function F7 cannot be satisfied by solution S7. This limitation is dealt with in a new model proposed in Section 4. In addition, the application of the Freeman and Newells' model is limited to designs using known structures in known combinations, i.e. *non-innovative* designs (see the discussion in Section 6).

2.2 Paradigm model

The paradigm model, at most, is a process of how a design could be done at a given level of detail. It proposes to do so by *modifying* an existing solution, which meets only part of the specification, by replacing those of its components which do not contribute or contribute negatively to satisfying the original specification, with new and more contributory ones. Now the remaining problem is identified as the difference between the original specification and the function provided by the new provisional solution. This solution is then modified by replacing its wrong components with new provisional components. However, these two modification processes (of problem and of solution) are not compatible. In order to be so, the problem reformulation should be done in terms of the difference between the original specification and the function provided by the provisional solution *without the wrong components*. Even with this modification in the model, there still are certain underlying assumptions which need to be true before this model could be applied. Firstly, the identification of *wrong components* in a provisional solution would require the relations between the function of the solution and that of its components to be known beforehand, such that the function of the provisional solution in the absence of the wrong components could be determined (we shall term this property *deducibility*). The second assumption is that an evaluation criterion for the *satisfaction* of a solution to a problem is available, with which to deduce whether or not the potential of the solution to solve the problem is increased after a modification (we shall call this property *evaluatability*). The third assumption is that it is possible to *monotonically* modify a provisional solution to satisfy given functional requirements of a problem, i.e. that an increase in the satisfaction value of a solution ensures a move towards the required state of the solution (this is what we call *monotonicity*). This assumption is not valid unless the solution contains

independent (clusters of) components which separately control various functions provided by the solution, such that a replacement of one such cluster would not affect the functionality of the rest of the solution. In that case, if we find a provisional solution having two independent clusters of components, one of which satisfies one of two functions required by the problem, and the other partly satisfies the other function, we could replace this latter cluster with a suitable one (assuming that this also does not affect the functioning of the rest of the solution). This process would lead to the required state of the solution. This property of a solution is termed here as *decomposability*. The model, with the suggested modification in its scheme, would work, for problem solving at a given level, for situations where the above assumptions are valid, which is not necessarily the case. Even when a provisional solution, having n components, in a knowledge base having m components, is *deducible*, *decomposable*, *monotonic* and *evaluatable*, there would be, in the worst case, $(2^n-2)(2^m-2)$ possible potential sets of wrong components (see Ref. 9 for further details) to be checked against the satisfaction criterion, at each stage of modification. In this case, the number of computations increases exponentially with n, and is therefore computationally expensive. Apart from these, the model does not solve the issue of elaboration of designs through different levels of detail, and is confined to known (provisional) solutions, i.e. *non-innovative* designs. The limitations imposed by the assumptions and the intermediate computational intensity could be overcome if the representation of solutions and the model were changed in the following way. In those cases where the above assumptions are valid, the solutions could be divided into their decomposable (clusters of) components, which then would qualify as provisional solutions in their own right. In those cases where the above assumptions are not valid, the solutions could be stored and used as a whole in problem solving. The process would then have to be changed into one in which provisional solutions, without modification, would be progressively added together to serve the purpose (which is what the scheme in Section 4 proposes), rather than progressively modifying an existing solution by replacing its parts to fit the function.

2.3 *The systematic model*

The systematic model makes no specific assumptions regarding whether only known solutions (i.e. structures) would be used in designing, and hence aims to support designs of any nature (i.e. *routine* to *innovative*). It is not clear whether it aims to support elaboration of the function-structure through the levels of design detail. However, as the model proposes to continue function-structure elaboration before looking for solutions to fulfil the sub-functions, it at least aims to support design at a given level of

detail. Moreover, it aims to produce the function-structure in a solution-neutral way.

As this model has the potential of having a wider scope of applicability, we specifically analyse its claims for their validity.

In this model, the idea of a solution-neutral elaboration of the function-structure plays the major role in the synthesis of solutions to given problems. This is because it is believed that the generation of a solution-neutral function-structure would allow the exploration of a wider solution-space and yet bring the problem state closer to solutions. The questions that immediately arise are:

- How do we elaborate a function-structure in a solution-neutral way?
- When do we stop elaborating it?
- How do we ensure that the above process takes the problem-state closer to its solutions?

The model proposes that we elaborate the function-structure using *simpler* functions. These *simpler* functions would have to be solution-neutral in order to produce solution-neutral function-structures, but what is a *simpler* function? Though *complexity* is defined by Pahl and Beitz[10] as the '... relative lack of transparency of the relationships between inputs and outputs, the relative intricacy of the necessary physical processes, and relatively larger number of assemblies and components involved ...', it is proposed that the elaboration of function-structure should be continued only '... until the search for a solution seems promising ...'[10]. In fact, '... if existing assemblies can be assigned directly as complex sub-functions, the subdivisions of function-structure can be discontinued at a fairly high level of complexity ...'[10]. It seems, from these suggestions, that the working definition of *simplicity* of a function, as is implicitly followed in the model, is the prospect of getting a solution which can satisfy that function. The other point is, the elaboration of *all* possible function-structures, in a *solution-neutral* way and not just one function-structure, is evidently the prerequisite to obtaining the optimum function-structure. The question is then, is there a finite number of solution-neutral function-structures to a given function so that one can hope to find the optimum (based on some criteria)?

The above questions can be amalgamated into the following:

Given a design problem as a desired function or a set of desired functions, and a known set of structures capable of satisfying specified functions, does the above model ensure the generation of those solutions to the above problem which are combinations of the given structures?

10 **Pahl, G and Beitz, W** *Engineering design* The Design Council, London (1984)

Let the known set S of structures form a set F_s of functions that these structures are able to provide. In order that a given problem (defined in terms of a desired function or functions) can be elaborated into only a *finite* number of function-structures (a pre-requisite to the generation of an optimum function-structure) expressible in terms of a set F of solution-neutral functions, the set of these functions has to be *finite*, *distinct* and *complete*. A set of functions is *finite* when the number of functions in the set is finite. A set of functions is *distinct* when there is no function in the set which can be expressed as a combination of the other functions in the set. A set of functions is *complete* when all possible function-structures and solutions can be expressed using the functions in the set alone.

Assuming that such a function-set F, say, can be found (attempts to find similar sets include work on the *generally valid functions* by Rodenacker[11], Roth[12], Koller[13], and Krumhauer[14]), the functional requirement of a given design problem can be expressed in terms of a finite set of function-structures using a subset of the above functions. As the function-set F is complete by definition, each function of the function-set F_s of known structures S should also be expressible in terms of (combinations of) elements of F.

Suppose this hypothetical function-set F is known. Then, given a problem, we can find all its function-structures (using elements of F). The question is, given a function-structure, and a set S of structures capable of providing a set F_s of functions, where both the problem and the solutions have their functions describable in terms of (combinations of) elements of F, can we derive, using the above model of functional reasoning, the solution-concepts (as combinations of elements of S), if any, which satisfy the function-structures?

The answer is, the model would be able to detect only those solution-concepts the function of at least one component-solution of which would map onto (i.e. fulfil) a single function of the function structure (Figure 4a). However, in general, there would be solution-concepts that satisfy the function-structure, the functions of at least one component-solution of which would map onto (i.e. fulfil) a chunk of the function-structure rather than a distinct component-function of the function-structure (Fig. 4b). This is referred to here as the problem of *partitioning*. These solutions would remain undetected by the model (see Ref. 9 for a detailed proof).

So, coming back to the central question asked at the beginning of the subsection, the conclusion is:

It is not possible to generate the function-structures in a solution-neutral way, and yet guarantee a movement of the problem state towards the solution-space. This implies

11 **Rodenacker, W G** *Methodisches Konstruieren. Konstruktionbücher, 27* Springer, Berlin (1970)
12 **Roth, K** 'Systematick der Maschinen und ihrer Mechanischen Elementaren Functionen' *Feinwerktechnik* Vol 74 (1970) 453–460
13 **Koller, R** *Eine algorithmisch-physikalisch orientierte Konstruktionsmethodik* Z VDI 115 (1973)
14 **Krumhauer, P** *Rechnerunterstützung für die Konzeptphase der Konstruktion* Dissertation TU D 83, Berlin (1974)

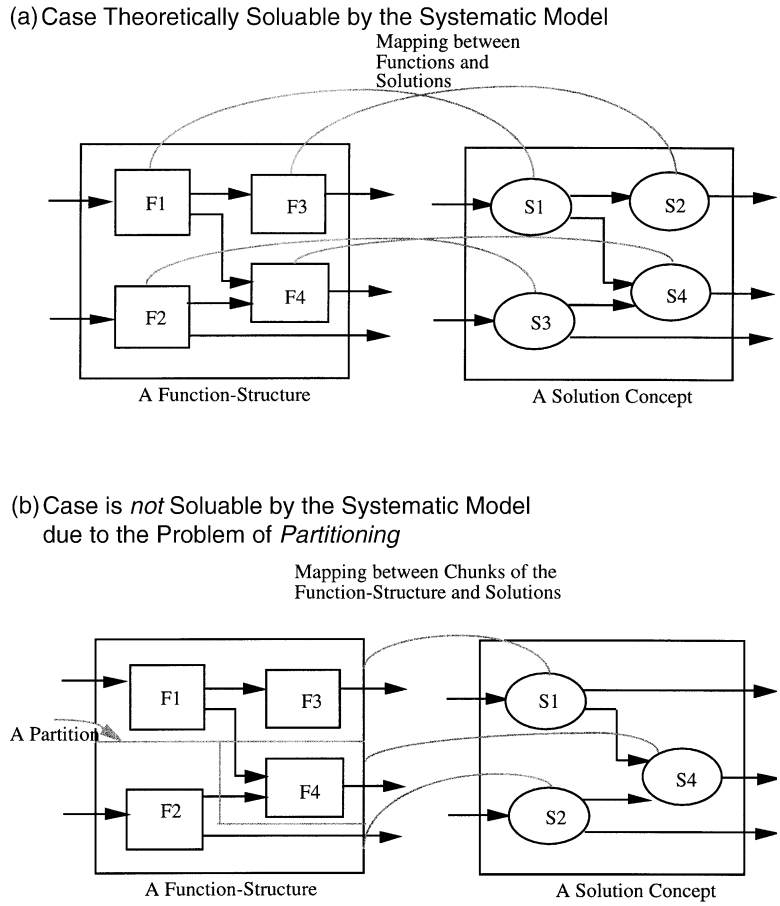


Figure 4 The systematic model and the problem of partitioning

F1, F2... : Sub-Functions in a Function-Structure
S1, S2... : Sub-Solutions in a Solution Concept

that the systematic model cannot be used to generate the function-structure in a useful way without being guided by the knowledge of existing solutions (see Note 1).

In actual situations, however, checking the *completeness* of a given set of functions would be difficult, if not impossible. Any function-set *F*, hypothesised as the *basic function-set*, would usually be incomplete. Consequently there would be, in all probability, a non-null set of known structures whose functions may not be expressible in terms of the functions in *F*. Given these problems, how can it still be ensured that at least all the solutions that are producible from the known structure-set *S* are generated?

We could get rid of the problem of design at a given level, associated with

15 **Chakrabarti, A** 'Supporting deep understanding and problem solving using function structures: a case study', in *Proceedings of the International Conference on Engineering Design 3, Tampere* (1997) pp 71–76

the model, by hypothesising the function-set F as a finite super-set of F_s, and then seeking for a framework to produce an exhaustive set of function-structures using F. In that case, the function-structures whose component-functions are subsets of F_s would be solvable, and the problem of *partitioning* would be eliminated. However, any function-structure which contains at least one component-function from the set $(F-F_s)$ would be unsolvable using the elements of set S. The fact that $(F-F_s)$ could be arbitrarily large (since in the absence of any criterion as to which functions can and cannot exist in the set, any function one could think of could be put into the set) and not contribute directly, if at all, to the generation of solutions, means that having any $(F-F_s)$ does not have any rational basis. We, therefore, make F equal to F_s, until a sound basis for keeping a non-null $(F-F_s)$ is found. This is equivalent to a model where given structures are combined to produce solution-concepts to given functions of a problem, and at least guarantees the production of solution-concepts that can be generated from the known set of structures.

3 *A proposed model for functional reasoning*

In this section, a new model for functional reasoning in design is proposed; this is based on the modifications suggested in the previous section and:

- represents structures or solutions using Freeman and Newell's[1] representation of structures in terms of the provided and required functions, and expresses these functions in a mathematical representation;
- guarantees the generation of solutions to a problem if there is a functional mapping between the known solutions and the problem;
- operates with a known set of solutions;
- provides a framework for design at a given level, which none of the existing models, except in some special situations, can provide;
- uses a framework, for elaboration of designs down through the levels, based on Freeman and Newell's model.

The model, shown in Figure 5a[16], is:

Given: a set of structures (known solutions), each of which has functions that it provides and those that it requires in order to provide each such function (exactly as in Freeman and Newell's model), and, a design problem, defined in terms of a (set of) function(s).

Required: structures (i.e. solution concepts), composed of the above structures, which would be able to solve the given design problem.

Scheme:

Step 1: The overall problem is first described as the initial problem definition (Fig. 5a). A part of this problem, defined by its function(s), is chosen for synthesis.

16 **Chakrabarti, A and Bligh, T P** 'Towards a decision-support framework for mechanical conceptual design', in *Proceedings of the International Conference on Engineering Design, 1, Zurich* (1991) pp 384–389

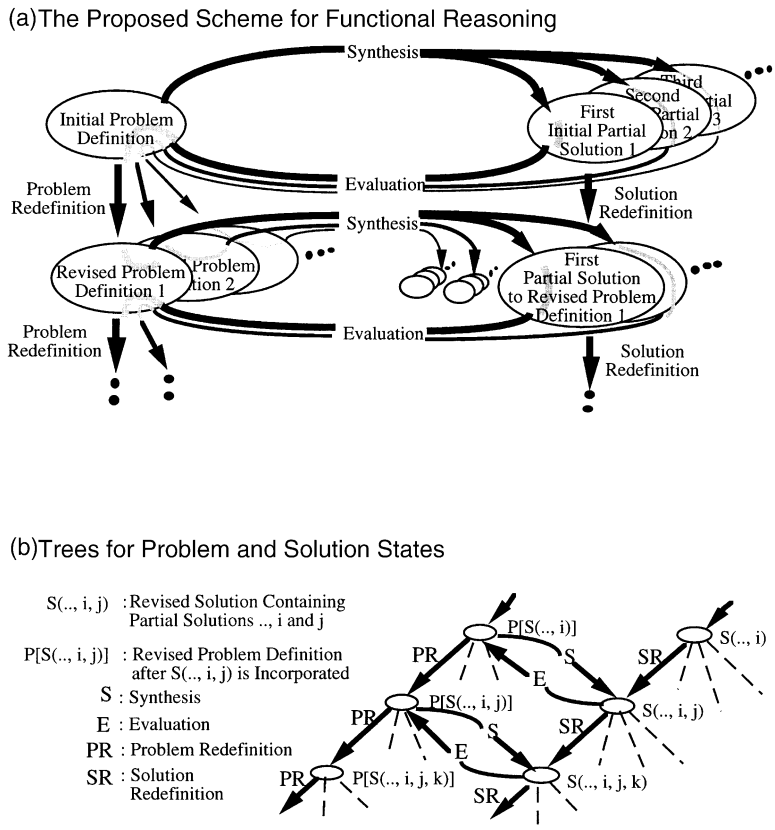


Figure 5 The proposed model for functional reasoning

Step 2: A set of alternative solutions is synthesised, from the known set of structures, to satisfy the functional requirements of the chosen part. These are shown as initial partial solution 1, 2, 3, etc.

Step 3: The function (i.e. the provided and the required functions) of the first alternative part-solution (i.e. first initial partial solution 1) is evaluated with respect to the functional requirements of the complete problem, which is then revised, incorporating the chosen part-solution. The provided function of the chosen part-solution reduces the extent of the functional requirements of the problem at the considered level of detail, while its required functions form the functions needed to be satisfied at some later level(s) of detail. The revised state of the problem is described as revised problem definition 1 in Fig. 5a.

Step 4: Alternative solutions are now synthesised for a part of the revised problem, the first of which (called first partial solution to revised problem definition 1) is evaluated to produce the next revised problem-function. This recursive process is continued until the problem is completely solved. The result of the steps so far is the generation of a single solution concept, which is an aggregrate of the partial sol-

utions chosen at each step of the synthesis-evaluation cycle (e.g. first initial partial solution 1, first partial solution to revised definition 1, etc).

Step 5: The above process has left behind a string of revised problem definitions, each resulting from the choice of a single partial solution from the alternatives generated at the previous steps. The search now goes back one step (i.e. choosing a partial solution different from that chosen previously in that step), revises the problem definition (Step 3), and solves for this revised problem function (Step 4).

Step 6: The search goes back to the step preceding that in step 5, and so on, until the solution tree (Fig. 5b) is searched completely (see Note 2).

The following are the essential features of this problem-solving model:

- The problem need not be solved as a whole, but can be tackled in parts.
- At every state of the problem, unlike in Yoshikawa's paradigm model, a complete partial solution is incorporated into the overall solution.
- The problem redefinition occurs in terms of the previous problem state (problem-function), the contribution of the newly chosen solution towards solving the problem (i.e. its provided functions), and the additional requirements the partial solution imposes (i.e. its required functions).
- The solution redefinition occurs by incorporating the present partial solution into the assembly of all the previous partial solution(s).

4 *A comparison of the functional reasoning models*

As stated in Section 2, a functional reasoning model ideally should support

- design of any nature, ranging from *routine* to *innovative*;
- design at a given level of detail;
- evolution of designs down through subsequent levels of detail.

The following two subsections explain the differences between and limitations of each of the previously described functional reasoning models, with an illustrating example.

4.1 *Limitations of and differences between the models*

Freeman and Newell's model[1] operates within the scope of known structures, and is therefore limited to supporting designs using known structures. It tracks down the design detail by identifying the functional requirements of the incorporated structures in a solution, and hence provides a way of elaborating the solution-concepts through subsequent levels of detail. However, it does not say anything about how to solve a problem at a

17 **Rutz, A** *Konstruieren als Gedanklicher Prozess* PhD Thesis, TU München (1985)

18 **Nidamarthi, S, Chakrabarti, A and Bligh, T P** 'The significance of co-evolving requirements and solutions in the design process', in *Proceedings of the International Conference on Engineering Design, 1, Tampere* (1997) pp 227–230

given level of detail, when there is no known structure which can solve the complete problem.

The paradigm model[2,3] also operates within the scope of knowledge of known provisional solutions, and is hence limited to supporting designs using known structures. It attempts to satisfy the functional requirements of a problem at a given level by progressive modification of an existing design by replacing its functionally harmful components by better ones. The scheme progresses by redefining the problem as the difference between the original functional requirements of the problem and the functions provided by the provisional solution, while solution redefinition takes place by incorporating only the useful part of the provisional solution, and later adding more useful components to it. These two redefinitions are incompatible, and in order that they be compatible, the problem redefinition should be the difference between the original functional specification of the problem, and that of the provisional solution less its wrong components. Even if this modification was incorporated into the model, it would ensure a solution to a problem at a given level only in the cases where the underlying assumptions of *deducibility*, *monotonicity*, *decomposability*, and *evaluatability* are valid, which is not the case in general. Moreover, the identification of *wrong components*, even when the assumptions are valid, would require computationally expensive (see Ref. 9 for further details), if not untenable, evaluation processes at each stage of the solution modification. Besides, it does not address the problem of evolving a solution through subsequent levels of detail.

The systematic model aims to support designs of any nature. However, it has been shown that unless the function-structure is built from the provided functions of known solutions (i.e. known structures), it cannot guarantee the generation of even those solution concepts which could be generated from the known solutions. Keeping any function which cannot be provided by known solutions in the set of functions used in forming the function-structures, leads to production of function-structures which are unsolvable in terms of the known solutions. This means that in order to operate effectively in solving design problems, the model has to be guided by the knowledge of known solutions. This makes its scope limited to the same as that of Freeman and Newell's. In that case, the model would be able to support synthesis of solutions to problems at a given level of detail. However, as the solution-specific elaboration of the problem is not done, unlike in the Freeman and Newell's model, the elaboration of the solution concepts is not supportable.

The proposed functional reasoning model (Figure 6) operates specifically within the scope of known solutions, and hence does not, as such, support

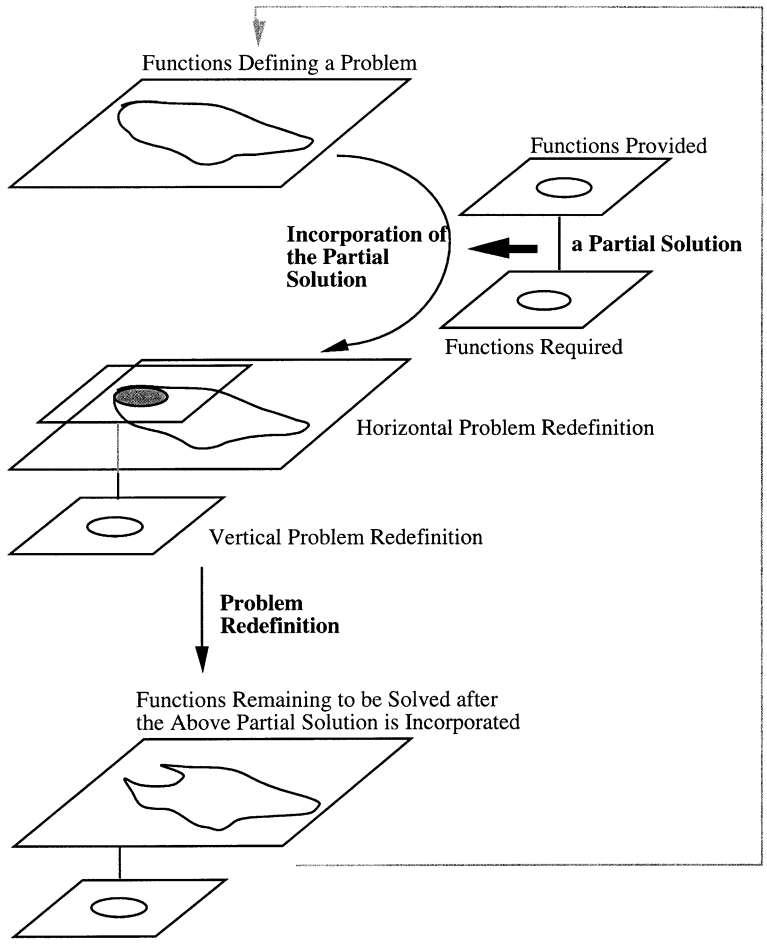


Figure 6 Vertical and *horizontal* problem redefinitions in the proposed scheme for functional reasoning

designs incorporating new design solutions. However, it supports the synthesis of solutions to a problem at a given level by recursive problem redefinition (we shall call this *horizontal redefinition*) while incorporating chosen partial solutions. Synthesis programs developed by Hoover and Rinderle[19], and Finger and Rinderle[20], for design of geared transmission systems, by Ulrich and Seering[21], for transducer designs, and, by Chakrabarti[22] for mechanical devices, could be taken as examples of *horizontal redefinition*. Horizontal redefinition allows the proposed scheme to support problem solving even if a given problem-function cannot be solved as a whole. Solving a problem at a given level using horizontal redefinition and in turn accounting for the functions required by the incorporated partial-solutions, in the revised problem definition (which is what we shall term *vertical redefinition*), allows the model to support the elaboration of the solution down through the subsequent levels of detail. The EDESYN program by

19 **Hoover, S P and Rinderle, J R** 'A synthesis strategy for mechanical devices' *Research in Engineering Design* Vol 1 (1989) 87–103

20 **Finger, S and Rinderle, J R** A transformational approach for mechanical design using a bond graph grammar in *EDRC report no. 24-23-90* Carnegie-Mellon University, USA (1990)

21 **Ulrich, K T and Seering, W P** 'Synthesis of schematic descriptions in mechanical design' *Research in Engineering Design* Vol 1 No 1 (1989) 3–18

22 **Chakrabarti, A** *Designing by functions*, PhD Thesis, University of Cambridge (1991)

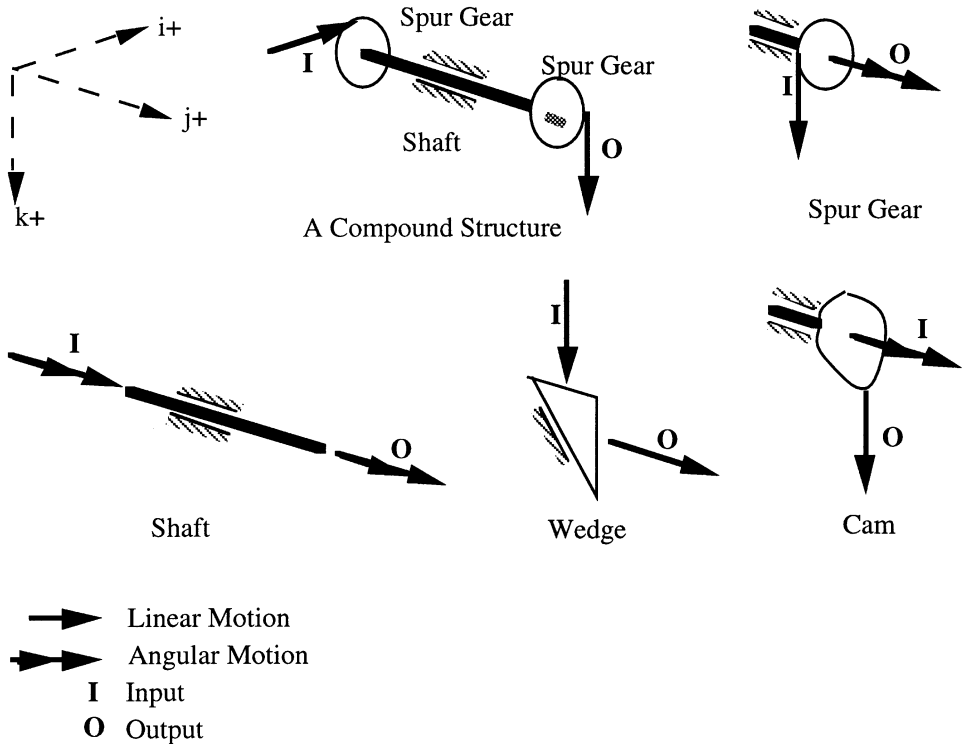


Figure 7 A knowledge-base of mechanical structures

Maher[23,24] for structural synthesis of multistoried buildings, in which pre-compiled planning knowledge (defined in terms of hierarchically arranged *frames* of *goals* and their *sub-goals*) is applied to identify the solutions which are attached as the *leaves* of the *goal frames*, could be taken as an instance of *vertical redefinition*. The proposed functional reasoning scheme, therefore, supports conceptual designs using known solutions at a given design level as well as down through levels of increasing detail.

4.2 An illustrative example

This example is to illustrate some of the problems associated with the former models for functional reasoning, and to demonstrate how the proposed model overcomes them. The problem solving considered is confined to a given level only (i.e. horizontal redefinition).

Suppose we have a knowledge base which contains (see Figure 7):

- a *compound structure*, consisting of two spur gears and an intermediate shaft connecting them. Its function is to transform an input force, transferred in through a gear–tooth interface, into an output force, transferred

23 **Maher, M L** 'Engineering design synthesis: a domain independent representation' *AI in Engineering Design, Analysis and Manufacturing* Vol 1 No 3 (1987) 207–213

24 **Maher, M L** 'Synthesis and evaluation of preliminary designs', in *Proceedings of the Fourth International Conference on the Application of AI in Engineering, Cambridge, UK* (1989)

out through a gear–tooth interface. The input and the output are on parallel planes;

- a *spur gear*, which can be used to take an input force through a gear–tooth interface, and transform it into a torque that is perpendicular to the force, or vice-versa;
- a *shaft* which would transmit a torque across its length;
- a *cam* which would take an input torque and produce an output force directed radially outwards, from the point of action of the torque, on a plane perpendicular to the direction of the torque; and
- a *wedge* which takes an input force through a contact interface and delivers an output force at a specified angle to the input, through another contact interface.

Let the design problem be (see Figure 8) the task of transforming an input force in a positive *i* direction, available through a tooth interface, into an output force in a positive *j* direction, to be transferred through a contact interface (this can be, among others, part of a door locking function). The position of the output is offset from that of the input in the *i*, *j* and *k* directions. Here *i*, *j* and *k* denote the principal spatial co-ordinates in a Cartesian system, and '+' or '−' denote the positive or negative sense of an orientation, respectively. This requirement can be specified as:

Input type: force	Input orientation: *i*
Input sense: +	Input interface: tooth
Output type: force	Output orientation: *j*
Output sense: +	Output interface: contact
Offset orientation-1: *i*	Offset sense-1: +
Offset orientation-2: *j*	Offset sense-2: +
Offset orientation-3: *k*	Offset sense-3: +

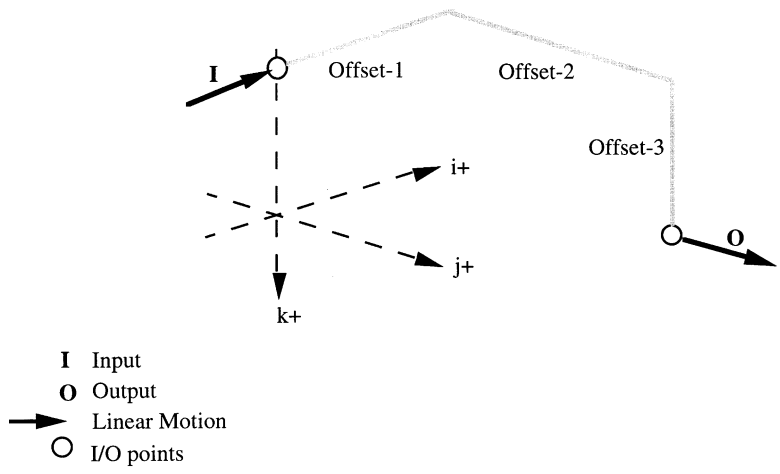


Figure 8 A design problem specification

Freeman and Newell's model would not be able to solve this problem, as it expects a single existing solution to match the functional requirements of the problem at the given level, which is not available in this case.

According to the paradigm model, the provisional solution that suits the specification best is chosen first. The compound structure is the choice, as it can satisfy all the requirements *except* the orientation, sense and interface requirements of the output force. The next step is to identify the *wrong components* (i.e. the ones which, if not present in the solution, would have made the solution better suited to solve the problem). If the output gear were deleted, as a potential wrong component, from the compound structure, the rest of the solution would satisfy all the requirements of the problem except the output type (which would now be a torque) and its sense, and, the offset orientation-2 and its sense requirement (which would be lost due to the removal of the gear). If the satisfaction criterion of a solution to a problem is based, in this case, solely on the number of parts of the requirement satisfied by the solution, the deletion of the spur gear would make the rest of the solution less satisfactory, and hence should not be pursued. If the criterion is based on assuming a hierarchical importance for various characteristics (such as: if I/O types are not matched, there is no point in matching any other characteristics; or, if orientation of a given type is not matched, matching the sense of the orientation does not mean anything; if type, orientation and sense of an I/O does not match, there is no importance in matching the interface requirements, etc.), even then the above deletion operation on the compound structure renders the solution less satisfactory than before. This is because, while the initial provisional solution could match both the input and output types, after the removal of the gear, it does not match the output type any more. Even if the modification strategy were taken as that of removing and replacing components at one go in the provisional solution (see Process 2 in Ref. 9 for more details), and only then checking to see if the satisfaction value of the modified solution has increased, and this were applied in this case by removing the gear from the compound structure and replacing it with a cam (as the best provisional component in this case), the satisfaction value of the resulting solution would not be increased. The resulting solution would then be incapable of meeting the output orientation, output sense, offset orientation-1 and offset sense-1, and therefore, according to either of the criteria defined above, would lead to a less satisfactory design than before. The model would, therefore, not get to a satisfactory solution. This is due to the fact that the model implicitly follows the *monotonicity* assumption (i.e. the more satisfactory a solution is, the closer it is to fulfilling the complete requirements), requiring the *decomposability* assumption (i.e. it is possible to play with the parts controlling the requirements which are

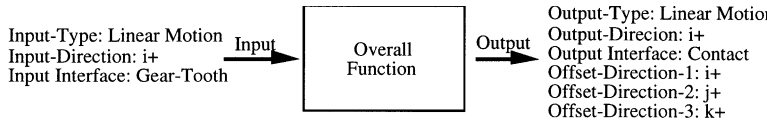
not matched by a provisional solution without adversely affecting some of the other requirements which are satisfied by the solution) to be valid, which is not true in this case. For instance, removing the output gear from the compound structure not only influences the unfulfilled requirements (i.e. the orientation, sense and the interface of the output), but also undoes some of the previously fulfilled requirements (i.e. the output type and the offset-1 requirements).

In order to use this example for evaluating the systematic model, we need to assume a set of functions, combinations of which are sufficient to represent the present problem and the available solutions. None of these functions, as a single entity, necessarily represents the function provided by an available solution. The presently chosen ones are:

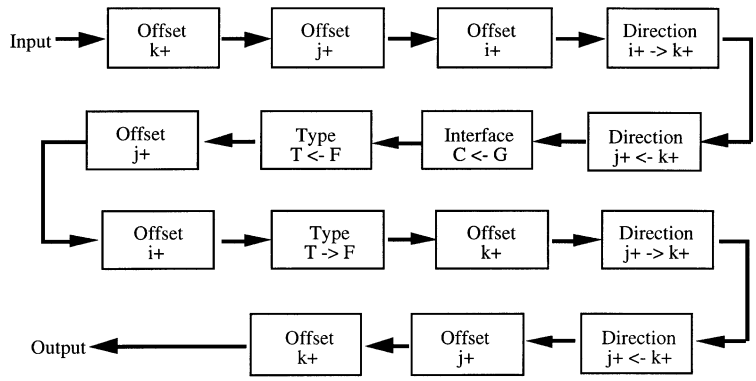
Function for type-transformation: input type=value→output type=value
Function for orientation-transformation: input orientation=value→output orientation=value
Function for sense-transformation: input sense=value→output sense=value
Function for offset-transformation: input offset=value→output offset=value
Function for interface-transformation: input interface=value→output interface=value

One of the ways the considered design problem can be expressed in terms of the above functions is the function-structure shown in Figure 9. If we could cluster the functions in the function-structure in the way shown in Figure 10 such that each cluster would map onto one of the available solutions, a solution to the problem could be found. However, dividing the function-structure into relevant chunks of functions, which would map into available solutions, is a combinatorially complex problem (described before as the problem of *partitioning*), which prevents the model from detecting the solution in this case.

In the case of the proposed model, the problem solving starts by choosing the compound structure as one of the partial solutions which at least matches the input requirements of the problem (i.e. type, orientation, sense and interface). The problem remaining, after the incorporation of this partial solution, is defined as the function of transforming the output of the incorporated structure, which becomes the input of the redefined problem, into the output of the original problem (see Figure 11b). A gear is one possible partial solution whose input can match the present problem input. The process of (horizontal) redefinition is done now as the transformation



(a) The Overall Function of the Problem of Figure 8



(b) A Function-Structure of the above Overall Function

Figure 9 The design problem of Fig. 8 and one of its function-structures, expressed in terms of a set of solution-neutral functions

Input or Output
T Angular Motion
F Linear Motion
C Contact Interface
G Gear-Tooth Interface
i, j, k Principal Spatial Orientations

between the output of the gear and the required output of the previous problem, and is shown in Fig. 11c. If this process is continued, one solution to the problem will be found after the fourth successive redefinition. This is shown in Fig. 11d, where the compound structure rotates the meshed spur gear, which in turn rotates, through the shaft, the cam. The cam pushes the wedge down, which then pushes through its other orthogonal edge the output in the required direction. The schematic diagram of the solution is shown in Figure 12.

In a detailed theoretical analysis[9], the best and worst-case estimates of the number of computations required in identifying *wrong* components in the paradigm model are developed, the average of which are compared with the worst-case estimates of the number of computations required in the proposed model. The results indicate that the proposed model has a better potential computational performance than the paradigm model.

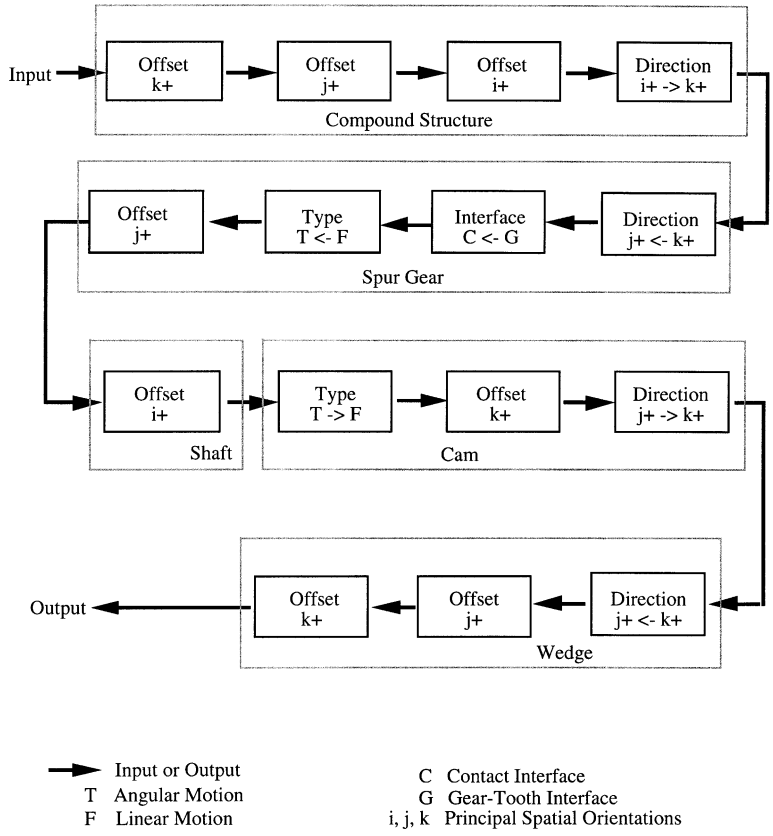


Figure 10 Clustering of the functions in the function-structure shown in Fig. 9b in the above way would have solved the problem of Fig. 9a

5 *Supporting innovative designs*

The proposed scheme for functional reasoning applies to engineering designs guided by the knowledge of existing designs. In the accepted terminology, this is *not* innovative or creative design. However, there is debate as to what is innovative or creative. According to Cagan and Agogino[25], innovative design is the process of deriving new design features from existing designs, and in creative design new primitives which have no obvious relationship to previous configurations are created. Maher and Gero[26] propose that innovative and creative designs are, respectively, the processes of *prototype adaptation* and *prototype creation*, where a *prototype* typifies a class of design, and thus serves as a generic design. According to Brown and Chandrasekaran[27], innovative and creative designs are different from what they call *routine* designs, where the structure of the artifact designed and the method of designing it are known. Ulrich and Seering[28] hypothesize that a large part of new designs are derived as novel combinations of existing designs. We propose that innovation and creativity lie as much in the new ideas (i.e. structures or solutions, in the present

25 Cagan, J and Agogino, A M 'Innovative design of mechanical structures from first principles' *AI in Engineering Design, Analysis and Manufacturing* Vol 1 No 3 (1987) 169–189

26 Maher, M L and Gero, J S 'Representing design knowledge as prototypes', in *Preprints of IFIP WG 5.2 Workshop on Intelligent CAD* (1987)

27 Brown, D C and Chandrasekaran, B 'Knowledge and control of a mechanical design expert system' *IEEE Computer* Vol 1 No 1 (1986) 92–100

28 Ulrich, K T and Seering, W P 'Computation and conceptual design' *Robotics and Computer-Integrated Manufacturing* Vol 4 Nos 3/4 (1988) 309–315

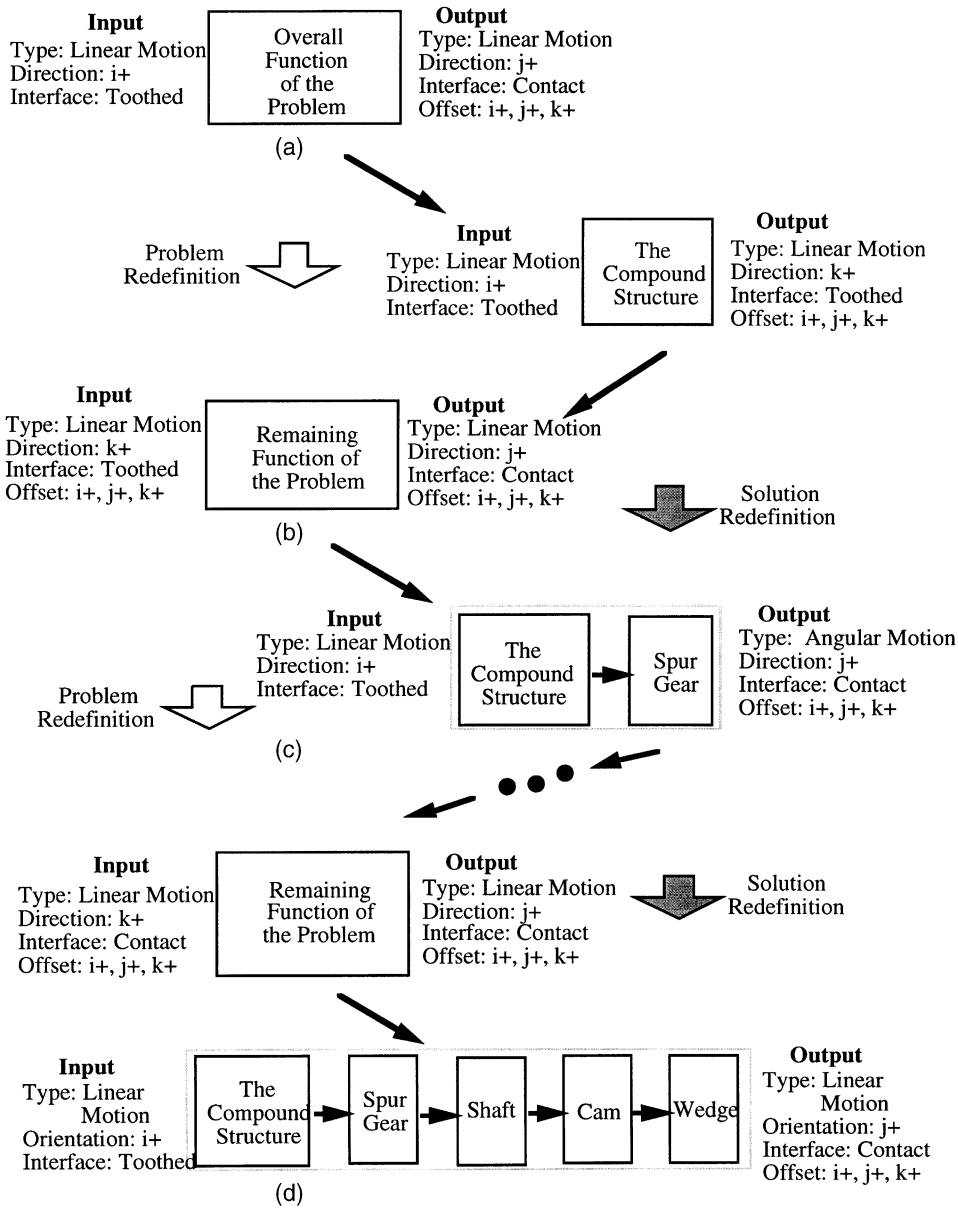


Figure 11 The proposed scheme proceeds through a recursive redefinition of the problem and solution to solve the problem in Fig. 10a

context) as in the ways in which these ideas are combined to form compound ideas or concepts. The proposed functional reasoning scheme is a framework for combining a set of known ideas/solutions/structures in all possible known ways to produce compound ideas/solution-

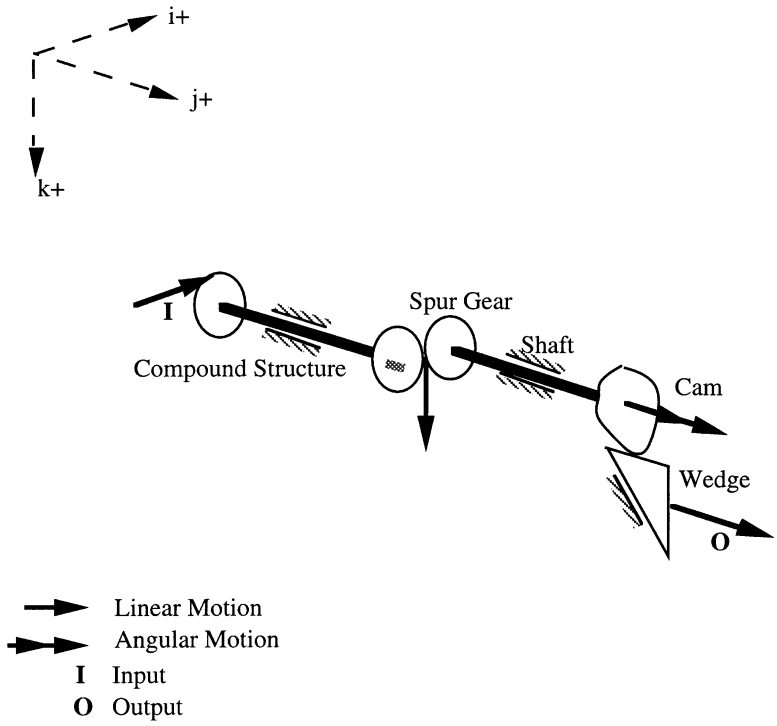


Figure 12 A solution to the problem specification in Figs. 8 and 9a

concepts/structures. The more *basic* the known solutions are and the more comprehensive their rules of combination, the more comprehensive and hence more innovative their combinations (i.e. solution-concepts generated) would be. Therefore, a parallel strand of research, examining the known solutions/structures, to discover the *basic* constituent elements and their rules of combination, should be established. Each time new original designs are found, these could be divided into either new *basic* solutions, or new rules of combination, or both, which would increase the repertoire of existing solutions and ways of combining them.

The model and this framework for extracting knowledge could support innovative design in at least two ways:

- when no acceptable solution to a given problem can be found using the existing *basic* structures and rules of combination, then one knows that new knowledge is required; and
- when a new solution to a given problem is found (which could not be produced using the existing ideas and combination rules), the new *basic* structures and rules of combination, of the new solution, extracted using the above framework, could then be used in the model to support a whole range of additional designs.

The research into identifying, extracting and validating *basic* elements and rules of combination from given solution concepts will be discussed in a further paper.

6 *Summary and conclusions*

In this paper, three existing functional reasoning models were reviewed, from the perspective of the three requirements of an ideal functional reasoning environment: one, the ability to support designs of any nature; two, the ability to support designing at any level; and, three, the ability to support evolution of designs through levels of detail. It has been found that none of these models support all of these requirements. Freeman and Newell's[1] model can support criterion three for designs using known structures; the paradigm model[2,3], with the suggested modification in its scheme, could support, in special situations, criterion two; and, the systematic model could, at best, support criterion two. None of the models support what is generally accepted to be innovative designs. It has been shown that a functional reasoning approach cannot guarantee the generation of solution concepts, which are combinations of known solutions, unless guided by the knowledge of existing solutions. A new model which can support design both across a level of detail and down through levels of detail (i.e. satisfy criteria two and three) has been proposed. Using a divide and rule approach and using recursive problem redefinition while incorporating existing solutions, the model supports conceptual design. The mathematical proof in Ref. 9 shows that the proposed model should have a better computational performance than the paradigm model. Although the generation of completely new solutions is not supported by the model, guaranteeing a search of the entire space of *known* solutions ensures that we know that *innovation* is required in the event of the model's failure to find an acceptable solution to a given problem. Each such *new* solution would lead to new structures and/or rules of combination being incorporated into the knowledge base, so that the model could then be used to support the design of classes of otherwise *innovative* designs. The idea is to have sustained progress towards transferring knowledge from existing designs in the form of *basic* structures and rules of combination, and thereby, increase the applicability of the model to designs which otherwise would be considered unsupportable in a systematic way. The proposed model is prescriptive in nature, and therefore does not claim to describe or explain human conceptual design process, although some similarity can be observed between its approach and descriptive findings[17,18].

Acknowledgements

Amaresh Chakrabarti wishes to acknowledge the Nehru Trust for Cambridge University, India, Cambridge Philosophical Society, Cambridge

University Engineering Department, the trustees of the Lundgren Fund of Cambridge University, the Cambridge Engineering Design Centre, the Northbrook Society, Darwin College, the Gilchrist Educational Trust, and the Leche Trust, for financial support. The authors acknowledge the helpful suggestions made by Rob Bracewell for improving the paper.

Notes

1 However, this limitation does not disqualify the use of the systematic model as a useful heuristic technique, which is what it was orginally intended to be, although some empirical studies[15] indicate that doing conceptual design by developing solution-neutral function structures was neither easy nor more beneficial than developing concepts without using them.

2 The descriptive design problem-solving model proposed by Rutz, on page 121 of his thesis[17], seems to provide some empirical underpinning of the model proposed here, although this is not necessary due to the prescriptive nature of the proposed model. The co-evolution of design problem and solution, which is an essential feature of the model, was also observed by Nidamarthi *et al.*[18] in their empirical studies of designers.