Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2005), *19*, 139–160. Printed in the USA.

DOI: 10.1017/S0890060405050110

A study on the grammatical construction of function structures

PRASANNA SRIDHARAN AND MATTHEW I. CAMPBELL

Automated Design Lab, Department of Mechanical Engineering, University of Texas at Austin, 1 University Station, C2200, Austin, Texas 78712-0292, USA

RECEIVED January 5, 2004; ACCEPTED January 25, 2005

Abstract

Function structures are used during conceptual engineering design to transform the customer requirements into specific functional tasks. Although they are usually constructed from a well-understood black-box description of an artifact, there is no clear approach or formal set of rules that guide the creation of function structures. To remedy the unclear formation of such structures and to provide the potential for automated reasoning of such structures, a graph grammar is developed and implemented. The grammar can be used by a designer to explore various solutions to a conceptual design problem. Furthermore, the grammar aids in disseminating engineering functional information and in teaching the function structure concept to untrained engineers. Thirty products are examined as a basis for developing the grammar rules, and the rules are implemented in an interactive user environment. Experiments with student engineers and with the automated creation of function structures validate the effectiveness of the grammar rules.

Keywords: Conceptual Design; Function Structures; Knowledge Representation; Shape Grammars; Tree Search

1. INTRODUCTION

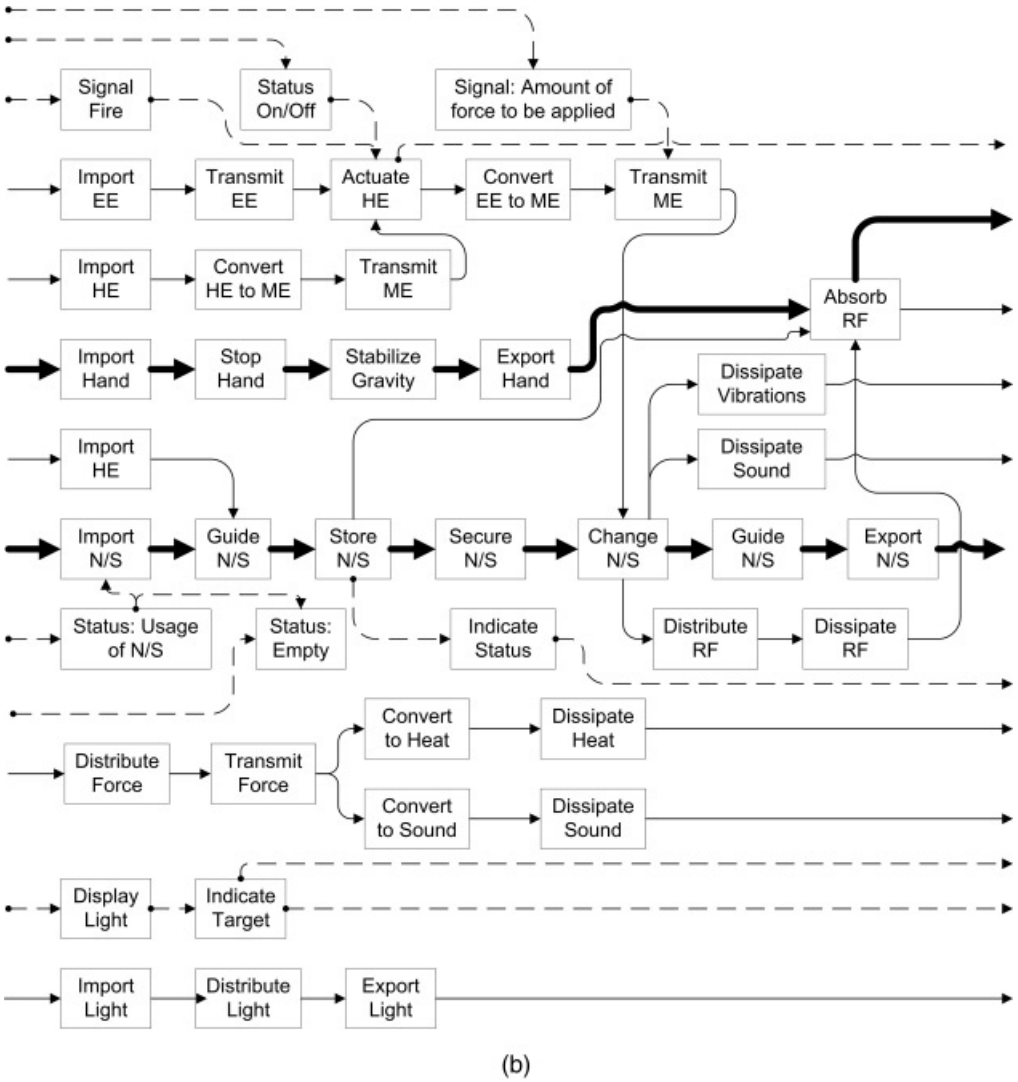
The function structure concept developed by Pahl and Beitz (1984) is a structured approach to managing complex conceptual design problems. There is no evidence that a design process taking advantage of this structured approach yields less creative solutions. In fact, applying function structures may provide a design team with more avenues for creative solutions than designing without such a technique. Function structures are significant because of their ability to help the designer tackle a design problem in a functional way rather than becoming overwhelmed by physical constraints early in the design process. In addition, function structures can remove the psychological bias that ties a design problem to previous solutions, and can be used to divide tasks among designers. The concept of separating form and function has been a useful heuristic for many complex engineering problems. Figure 1 shows an example of an artifact and its accompanying function structure. The Arrow Staple and Nail Gun has several key energy and material flows that flow through this product. By drawing a function structure, as in Figure 1b, a designer can approach the problem in smaller solvable pieces.

Typically, when designers use function structures, they end up making only one function structure. During this generation process, there are many instances where the designer makes unconscious and conscious decisions, thereby limiting himself or herself from the theoretically infinite solutions that are possible. Ultimately, the result is a single function structure that suggests an operating procedure of the product according to the satisfaction of the designer.

However, this is arguably not the best way of designing because these decisions prematurely narrow down the final solution and may miss out on other potential solutions that might give the designer new or better ideas to solve the problem. Consider Figure 2, where a tree search is shown in the context of function structures. As the designer makes decisions, various paths are traversed that finally takes us to a solution. However, in the infinite solution space, there are numerous feasible solutions, and generating all these feasible solutions is a tedious job for the designer. Compu-

Reprint requests to: Matthew I. Campbell, Automated Design Lab, Department of Mechanical Engineering, University of Texas at Austin, 1 University Station, C2200, Austin, TX 78712-0292, USA. E-mail: mc1@mail.utexas.edu

(a)



(b)

Fig. 1. A Black & Decker electric knife and its function structure.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

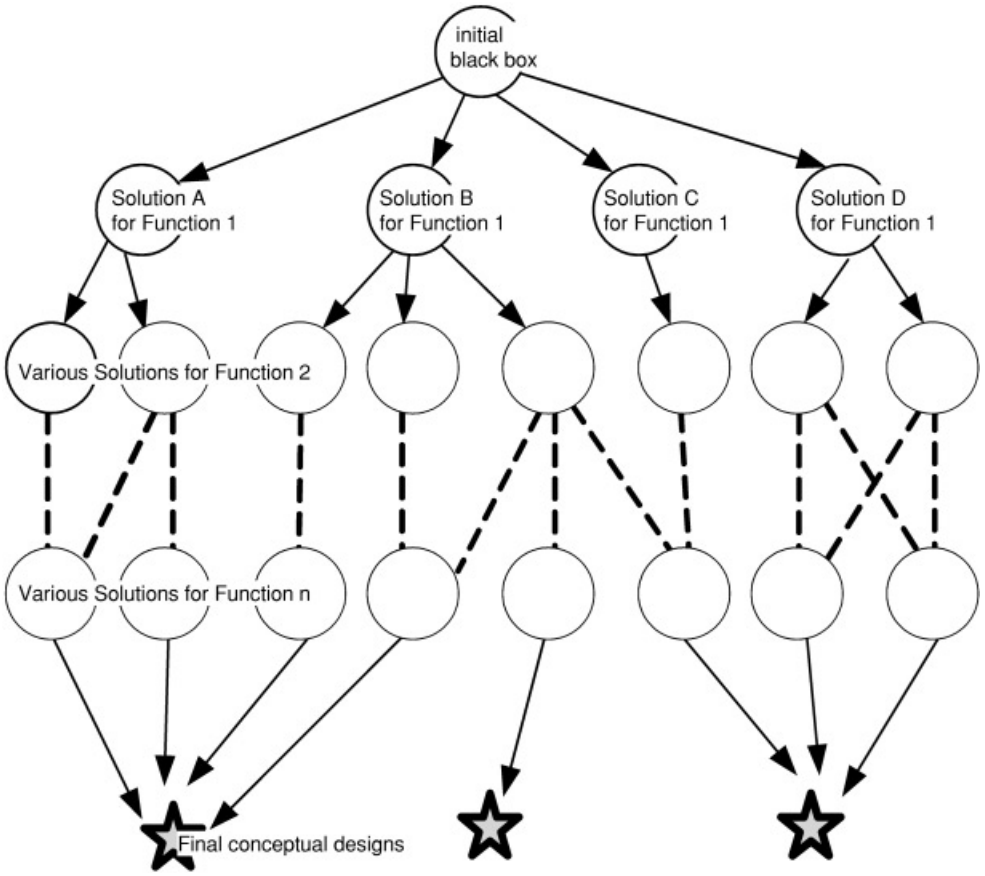


Fig. 2. A designer committing to different decisions about each function is analogous to a search through a decision tree where the goal is a conceptual design.

tationally generating the set of "feasible" function structures would allow the user to identify new conceptual designs or build from various concepts to create a completely different, yet novel solution.

This paper presents an approach to representing this tree by a formal language for functions. Based on the common basis of function names developed by Stone and Wood (2000), 69 grammar rules are created to capture the feasible set of interactions between these functions. As an analogy to written language, the common basis provided in Stone and Wood (2000) represents the *lexicon* of valid terms, while our approach provides the *grammar* for combining these terms. The main reason for developing these grammar rules is to provide a framework that can be used to generate function structures. Once the framework is established, it can be used to generate multiple function structures.

Although function structures are flexible and powerful to use, they are still not widely utilized in industry. This can be attributed to the fact that there are inherent difficulties associated with teaching function structures and conveying their meaning between designers. This is a consequence of the degree of variability in what exactly constitutes a valid function structure. Creating function structures tends to be an "internal" activity where either an individual designer or a group of designers create the structure as the basis for brainstorming or to divide the design problem into subsystems. Most of the variability comes from how much detail one strives to include in the function structure. A lack of a fixed granularity can make two function structures for the same artifact appear completely different. If formal guidelines can be provided for creating function structures, they can overcome the problems of correctness and granularity, and thus will be easier to share among different groups of designers and easier to teach to new designers.

The set of developed grammar rules lends itself well to computer implementation. A computer implementation makes possible the generation of numerous function structures for a single product that can then be combined or modified by the designer based on his/her own experience. Because recent grammar research has been implemented at various levels, Chase (1998) developed a model to characterize different implementation scenarios. In Figure 3, six scenarios are presented that are ordered by the amount of human interaction versus computer interaction. The current function structure grammar is most useful under Scenario 4, in which the computer is able to recognize which

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

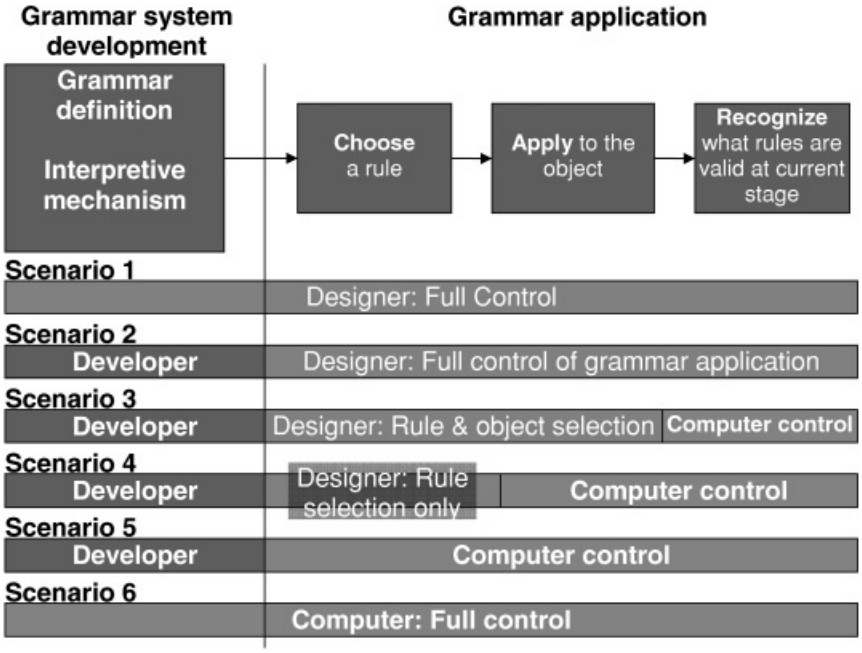


Fig. 3. Chase's six scenarios for classifying grammars (adapted from Chase, 1998).

rules are applicable and apply rules to update the design. The designer can use this tool for creating function structures quickly and accurately. The user interacts with the system to choose the rules to apply in order to build a complete design. By having a computational search strategy traverse the tree, we achieve a Scenario 5 grammar under Chase's model. The automatic creation of function structures provides the user with a brainstorming tool to develop new ways to solve the basic functions of a design. For example, a random choice of valid grammar rules might create alternatives to the staple gun function structure in Figure 1. Alternative energy sources other than electrical energy could be used to punch nails and staples. Having a computer choose rules also breaks new ground in that the automated approach would reason with and establish functional conceptual designs independent of a user.

In this paper, we present our approach for developing the graph grammar for function structures. This includes the study of past function structure methods, past graph grammar approaches (Section 2), and an empirical study of existing artifacts (Section 3). Our resulting set of 69 rules is then summarized (Section 4) and their operations are shown through an illustrative example (Section 5). Results from a human-based study (Section 7.1) and a computational study (Section 7.2) are also discussed.

2. RELATED WORK

Function structure research has found its way into a number of educational texts (Ulrich & Eppinger, 1995; Otto & Wood, 2001) because of the presentation provided by Pahl and Beitz (1984). Furthermore, research building upon the concept of function structures has flourished in the past 15 years. Numerous publications have extended the application of function structures (Collins et al., 1976; IEEE, 1998; Chen et al., 2002) and formalized the use of such structures (Hubka et al., 1988; Kirschman & Fadel, 1998; Szykman et al., 2000). Computational approaches have also been explored that further expand the value of function structures (Gietka et al., 2002; Wang & Yan, 2002). In a way, this detailed research into how to represent function is a subset in a larger design process where solutions to function precede those of an artifact's structure. The function–behavior–structure approach presented in Gero and Kannengiesser (2003) has provided a theoretical approach to managing design knowledge as it is created by a team of engineering designers. Although the function structure approach discussed here has often been developed "top-down" from how engineers divide a larger design problem into manageable elements, continuing efforts (including those discussed here) are closing the gap with the related work of functional representation in artificial intelligence research (Chandrasekaran et al., 1993).

In recent years, engineering researchers have discovered that shape grammars, originally used in architectural research (Stiny, 1980), provide a flexible yet ideally structured approach for various aspects of engineering design (Cagan, 2001). A shape grammar is a set of shape rules that are applied in a step by step way to generate a set, or language, of designs. Grammar-based design systems offer the option of exploring design alternatives as well as automating the design generation process. An experienced designer can construct a set of rules to capture his/her knowledge about a certain type of artifact. The grammar can then be con-

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

structed such that any execution of the rules creates a feasible solution (Longenecker & Fitzhorn, 1991) or captures the style of a specific period (Cagdas, 1996) or of a specific designer (Koning & Eizenberg, 1981).

An important offshoot of the shape grammar research is graph grammar research. Similar to production systems in cognitive psychology (Klahr et al., 1987), graph grammars are comprised of rules for transforming nodes and arcs within a graph. These techniques create a formal language for generating and updating complex designs from a simple initial specification or seed. Graph grammars are an emerging concept in design synthesis (Pinilla et al., 1989; Fu et al., 1993; Schmidt, 1995; Li et al., 2001). The development of these rules encapsulate a set a valid operations that can occur in the development of a design. Such representations can produce a wider variety of candidates because solutions need not have common characteristics, but merely a common starting point.

According to Kurfman et al. (2001), "the problem with many of these function-based design methodologies is their inability to produce repeatable functional models of a particular product. Two engineers can be given the same product, customer needs, and the process choices, but the likelihood of them producing similar function structures is low." This is a major issue in using function structures. To develop a formalism for function structures, there needs to be a common language in which the formalism can be developed. The basis that was developed by Stone and Wood (2000) is used for this purpose. The development of the functional basis helps to foster our understanding of function structures and aids in developing repeatable function structures. Hirtz et al. (2001) developed the reconciled functional basis, which covers most of the functions being used in engineering artifacts. This basis consists of eight primary function classes and three flow classes that are further subdivided into secondary and tertiary functions and flows, respectively. This basis has been adapted to suit our formalism and requirements.

3. APPROACH

The typical functional structure methodology starts with a black-box model representation of a product where the identified inputs and outputs to the system are shown on the left and right side of the boxes, respectively, and the primary function of the product is shown within the box as a verb–object pair. Our approach begins by developing a chain of subfunctions that operate on each of the input flows. Consider the example of an Electric Staple/Nail Gun. On observing the product, we notice that the staples and nails are guided into the product and are secured using a spring. We see that the primary function of the gun is to shoot the staples or nails, which have been previously inputted and secured. To shoot, we need to impart some form of energy in the device. It is seen that the energy that is used here is mechanical energy, which is obtained by converting the incoming electrical energy. Hence, we need to import electrical energy and convert it into mechanical energy. In addition, we need human energy to direct when to actuate the electrical energy. Human energy enters the system and gets converted to mechanical energy due to the movement of the hand and this energy actuates the system. We also need to model the staples and the electrical energy as these interact with each other. Following this systematic dissection, we can create all the functional chains for all of the flows entering and leaving the black-box model. This analysis reduces the final product to its original functional specifications. Our goal, however, is to be able to aid the engineer before the design is complete; therefore, we seek a method to automatically generate multiple function structures given the black box as a seed or starting point.

This paper presents the preliminary findings toward this goal. The hypothesis of this work is as follows: a formal set of rules can be created that will describe function structures for a range of products, based on a common basis, and which, when executed, will help in the automatic generation of function structures. Derived from this hypothesis are certain objectives that need to be satisfied. It needs to be shown that a set of rules can be developed that can effectively describe function structures. This set of rules then can be programmed as an algorithm that can be used to generate function structures. Thirty consumer products are chosen to provide an empirical basis for creating our rules. These products offer simple technologies, yet the range of technologies that each of these products uses varies enough to present a challenging problem. The products that were examined are shown in the Table 1.

One of the first challenges we faced in comparing the functions structures created for these products is the unpredictability of signal flows. Signal flow depends to a large extent on the environment in which a product is used. We

Table 1. *The 30 products used as an empirical basis for the development of the grammar rules*

Products Chosen	
Rice cooker	Electric iron
Boston Hunt sharpener	B&D electric knife
Room heater	Krups coffee maker
Coffee grinder	Electric cordless kettle
Electric toaster	Leaf blower
Electric wok	Popcorn popper
Dustbuster	Skil screwdriver
Water pump	Hair dryer
Kenmore dryer	Floor jack
Spatula mixer	Coleman quick pump
Stapler	Jigsaw
Dremel engraver	Fruit and veggie peeler
Presto Salad Shooter	Ball shooter
Palm sander	Cordless drill
Hand blender	Juicer

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

believe that trying to model signal flows is cumbersome and leads to rules that are too vague and general. Hence, signal flows have not been considered at this time.

Another challenge in our study was determining the level of granularity to use in systematically making function structures. In the functional basis (Stone & Wood, 2000) three levels are developed: primary, secondary, and tertiary. Using different levels of granularity leads to different function chains. This poses a problem for formalizing the grammar for function structures. The problem becomes a lot more complex if the developed system of rules must differentiate and understand the three levels. This necessitates the formation of a new reduced basis that incorporates the elements of the previous functional basis. For example, the secondary level of *Guide* has four different words at the tertiary level: *Guide*, *Translate*, *Rotate*, and *Allow DOF*. To create a grammar that is consistent, we have to create a rule that would apply to secondary class Guide, that would be applicable to the tertiary class as well. At the same time, because these bases have their own intricate differences, rules have to be created to maintain those differences at the tertiary level. This creates a complexity that becomes magnified when all the primary, secondary, and tertiary classes are taken into account. It is a much better approach to create the rules considering just one level, as this will result in a more robust set of rules. The reduced functional basis is shown in Table 2.

Table 2. *The three levels of granularity in the functional basis and our reduced single level used in our function structures*

Primary	Secondary	Tertiary	Modified Functional Basis
Branch	Separate		Separate
		Divide	Divide
		Extract	Extract
		Remove	Remove
	Distribute		Distribute
Channel	Import		Import
	Export		Export
	Transfer	Transport	Transport/transmit
		Transmit	
	Guide	Translate	Guide
		Rotate	
		Allow DOF	
Connect	Couple	Join	Couple
		Link	
	Mix		Mix
Control	Actuate		Actuate
Magnitude	Regulate	Increase	Regulate
		Decrease	
	Change	Increment	Increase/decrease
		Decrement	
		Shape	Shape
		Condition	Condition
	Stop		Stop
		Prevent	Prevent
		Inhibit	Inhibit
Convert	Convert		Convert
Provision	Store	Contain	Store
		Collect	
	Supply		Supply
Support	Stabilize		Secure
	Secure		
	Position		Position

An empirical study is conducted to extract rules from function structures of different products. This study culminated in the design knowledge that is captured in the developed rules, which are implemented to generate new functional models. We observed similar rules or patterns from the function structures created for products shown in Table 1. By using the reduced basis, we are able to more readily develop generic rules that are valid across all products. Because the same rules were seen in successive products that were examined, the number of newly created rules diminished throughout the process. Figure 4 shows the graph between products examined and the rules obtained from them. The number of rules appears to asymptote toward a finite number. This validates our assumption that a finite set of rules can describe all function structures in our current product domain. We recognize that the rules that have been developed do not fully represent the entire set of engineering artifacts, but the current set provides a basic framework on which future rules can be developed.

Grammar-based design systems offer the options of exploring the design alternatives as well as automating the design generation process. It may be argued that using grammars can hinder the creativity of the user as the generated solutions are based on a predefined set of rules, which makes the generated design repetitive and clichéd. However, the use of a design grammar helps to generate a wide range of solutions by altering the way the rules are applied and changing the rules themselves with little or no input from the user. This can give the designer the potential to evaluate a large number of alternative designs without tedious work. This set of designs generally includes many alternatives that might have been overlooked by a designer who is working without the aid of a grammar, thus paving the way for possible innovative designs. The shape grammar rules can be used to represent the transitions between states in a search tree as shown in Figure 2. Thus, such a representation of a design space requires navigation techniques to enable search for a desired or optimal solution. The issue in implementing the grammar now becomes one of controlled searches through the space of solutions. In our case, we start with the black-box model as the initial seed. Each rule that is recognized as feasible opens up different avenues that can be explored. We have the option of applying any of those rules and then once again evaluating the design to see where, which, and how many rules can be applied. This process is repeated until there are no more rules that can be applied. The logic that is followed for recognizing, applying, and reevaluating the design at each stage is explained in the next section of this paper.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

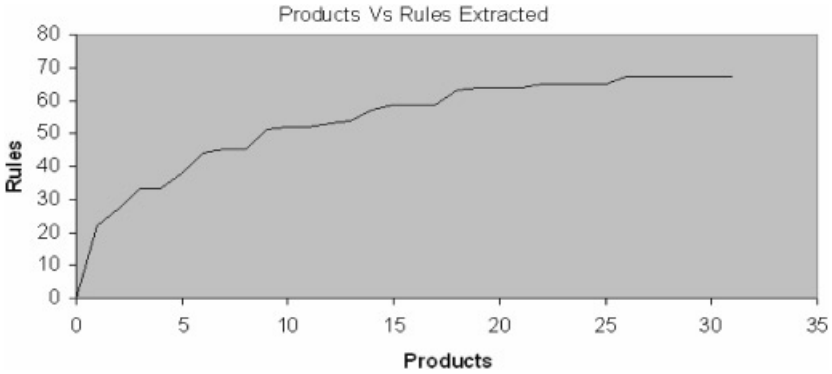


Fig. 4. A graph of the Number of Products examined versus the Rules obtained from each of them. The products are listed in random order.

4. RULE SET

In this section, we shall discuss the rule set that is developed and the methods that we use to generate the rules. A function structure is composed of two basic units: functions and flows. A function can be defined as "a description of an operation to be performed by a device or artifact, expressed as the action verb of a function block." A flow can be defined as "a change in material, energy or signal with respect to time that is expressed as the object of a function block, a flow is the recipient of the function's operation" (Stone & Wood, 2000). The main types of flows are shown in Table 3.

A major issue in the implementation of these rules is the recognition as to where and when these rules can be applied. To this end, we have borrowed the concept of active centers from the phenomenon of polymerization. In polymerization, chemical agents operate on distinct locations on a polymer chain (Progelhof, 1993). These locations are called active centers because they are potential areas of attachment by incoming molecules. Similarly, during the creation of a function structure, there are many "active centers" where incoming flows and functions can attach themselves. These active centers are the points where grammar rules can be applied and where new functions and/or flows are added if certain criteria are met at a specific open connection. In addition to borrowing the active center concept, the creation of a function structure follows a similar initiation, propagation, and termination process as that of polymerization. Rules are thus divided into three groups based on whether they create, maintain, or consume active centers: initiation rules, propagation rules, and termination rules. The *initiation* rules are first recognized and applied until no more rules from this group are recognized. This stage of the process includes rules that tend to create more active centers than they consume. The *propagation* rule set is then used to check for active centers and these rules are similarly applied until no more grammar rules are recognized from this group. As in polymerization, propagation tends to maintain a constant number of active centers. The *termination* set completes the function structure generation process by invoking rules that eliminate the remaining active centers.

Table 3. *Common basis of energy and material flows*

Energy	Material
Human	Human
Acoustic	Gas
Biological	Liquid
Chemical	Solid
Electrical	Mixtures
Electromagnetic	Container
Hydraulic	Tool
Magnetic	Battery
Mechanical	Air
Pneumatic	Water
Radioactive/nuclear	Ice
Thermal	Steam
Reaction forces	

The current development of rules has been modeled after the basic grammar conventions where rules contain both a left-hand side and a right-hand side. The left-hand side contains the state that must be recognized in the current configuration and the right-hand side shows how the design is updated to a new configuration. As long as there are "active" centers, then the function structure is not complete. At any instant there are a number of rules that can be applied, and the implementation of the grammar recognizes all of these possibilities and the associated active centers to which they apply. The rule that is actually applied can be a choice of the user or an automated process. The choices determine which branch of the tree to follow in reaching the final state of the generated function structure.

Some of the basic rules are shown in Figure 5. The gray circles with the black dots inside them that are present in these figures represent active centers, or the potential areas where flows and functions can be added. In Figure 5a, we

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

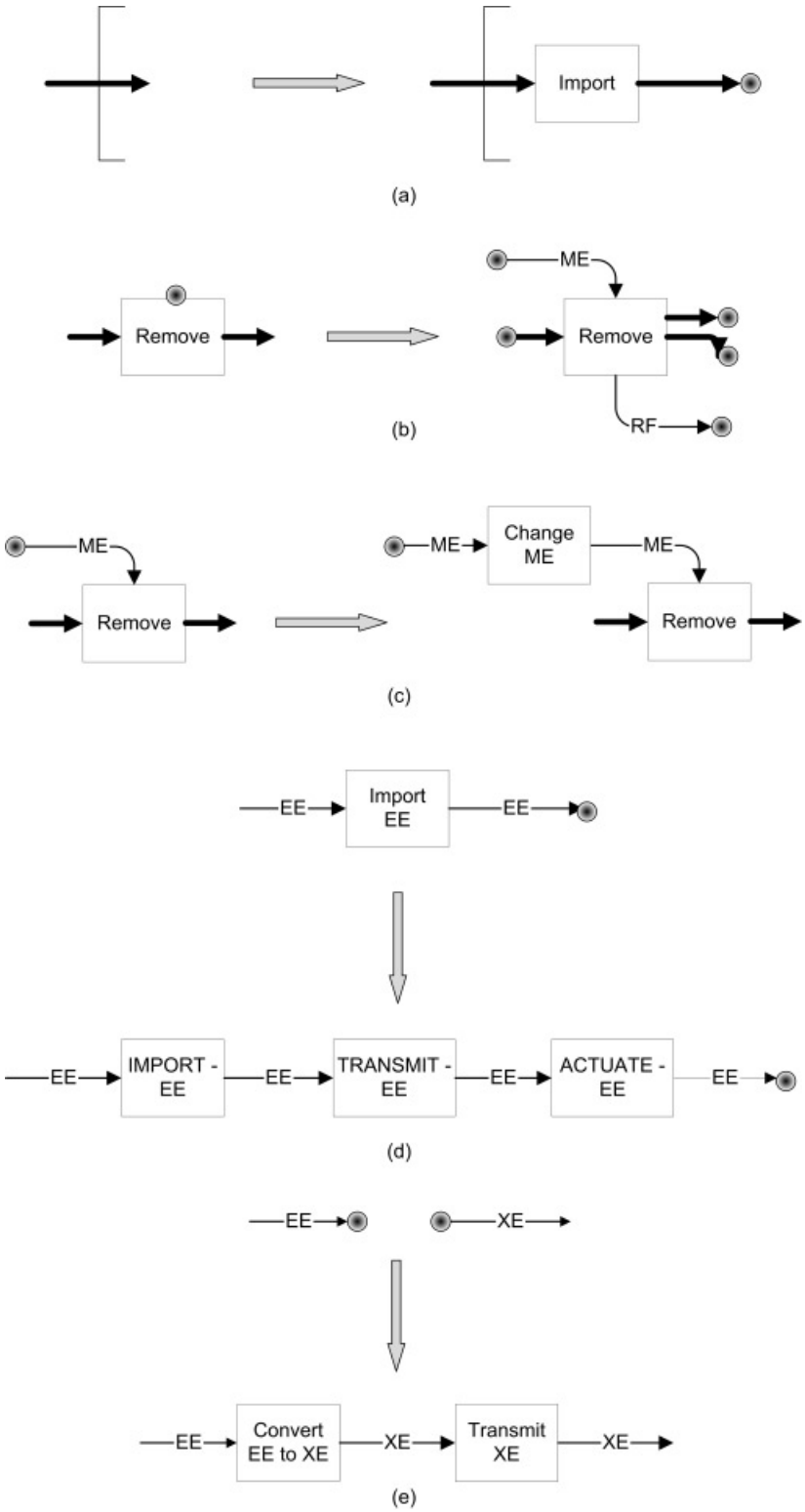


Fig. 5. Five of the 69 grammar rules developed for creating feasible function structures.

see an active center being created. This rule recognizes any flow that is coming into the black box. If such a flow is recognized, it adds the function "Import" to the head of the flow and adds another open flow of the same type in front of the function. This is an example of an initiation rule.

Consider the rule shown in Figure 5b. This is an example of a rule where the active center is on a function rather than

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

on a flow. It recognizes a function named "Remove" and adds a mechanical energy flow as the secondary flow to its back and a reaction force flow to the front of the function. It also replaces the solid coming out of the flow with two solids. This rule captures the principle that whenever we cut or grind a solid, we need to supply some mechanical energy and this results in two or more pieces of that solid. It should also be noted that this rule results in the creation of many active centers. This rule cannot be applied again because the active center necessary for rule recognition has been eliminated. Care must be taken to define rules that prevent the same rule from being applied over and over again.

The rule shown in Figure 5c captures another common principle. Usually, when mechanical energy is being supplied, the energy is amplified using gears and this is represented by the function "Change ME." This rule looks for a flow of type mechanical energy that is open at the tail and is pointing to the function, "Remove Solid." If applied, this rule adds the function Change ME to the tail and adds another flow open at its tail to the back of the Change ME function. Figure 5d shows a rule where an electric energy flow that is pointing from Import is recognized and the functions "Transmit EE" and "Actuate EE" are added to it. This rule is observed in many products that use electrical energy, because electrical energy is always transmitted and actuated before being converted to the required form. The above two rules each remove an active center but end up creating a new one. The rules shown in Figure 5b–d are propagation rules as they use and replace active centers.

The final rule shown in Figure 5e is a termination rule where active centers are removed. Whenever two flows are recognized such that we have an open electrical energy flow and energy of any other kind (represented as XE) that needs to be supplied, we convert the electrical energy to the required form and transmit it. Termination rules are vital in obtaining a valid function structure. In the next section, we will illustrate and discuss how these rules are used to create an example function structure.

5. ILLUSTRATIVE EXAMPLE

The Black & Decker electric knife is used to illustrate how rules are recognized and applied to construct a valid function structure. As seen from the black box in Figure 6a, the primary function of this product is "Cut Food." The black box is then rephrased into the language of the common basis, which in this case is Remove Solid. Traditionally, the black-box model contains a single verb–noun description of the intended function of the product and the input and output energy, material, and signal flows. Figure 6b shows the black box in the common basis language that we use in the grammar.

Figures 7, 8, and 9 present snapshots of the generation process of the three stages of initiation, propagation, and termination. Four snapshots between start to finish of each stage has been provided. The active centers overlaid with colored squares show the active center that has been selected for rule application. The ovals show the result of rules applied at a particular active center. In between the snapshots, we list the grammar rules that are applied (for a complete description of all 69 grammar rules, see www.me.utexas.edu/~adl/fs_grammar.htm).

Consider Figure 7. The algorithm first reads the black box and creates an empty function structure that simply has the inputs, outputs, and the primary function. At this stage,

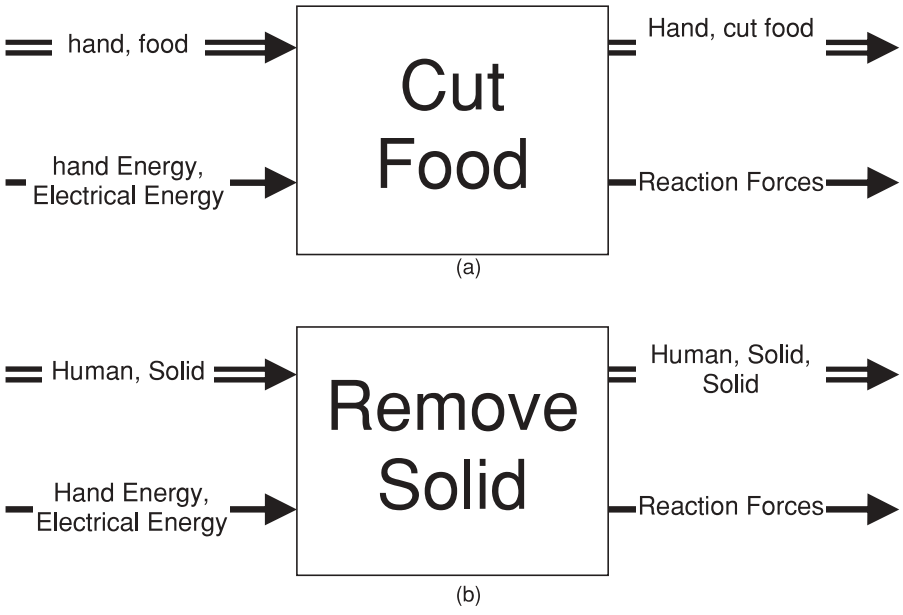


Fig. 6. (a) Electric knife black box and (b) black box in reduced basis.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

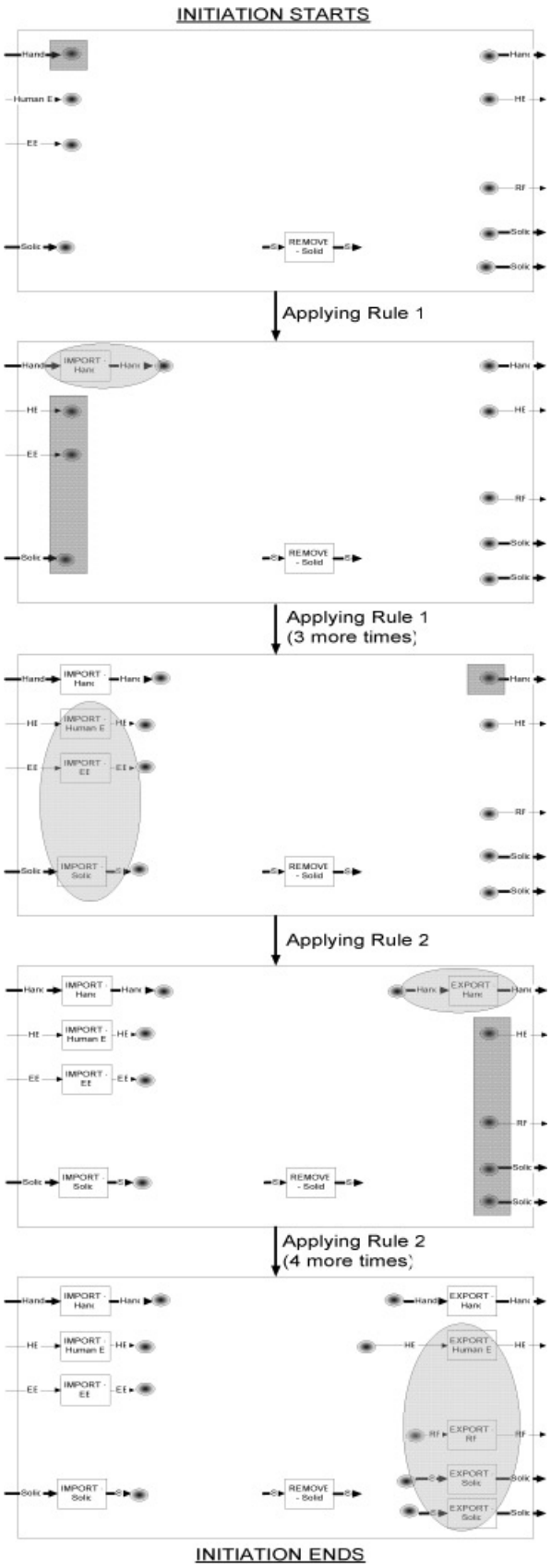


Fig. 7. A pictorial representation of the generation of the function structure in the initiation stage.

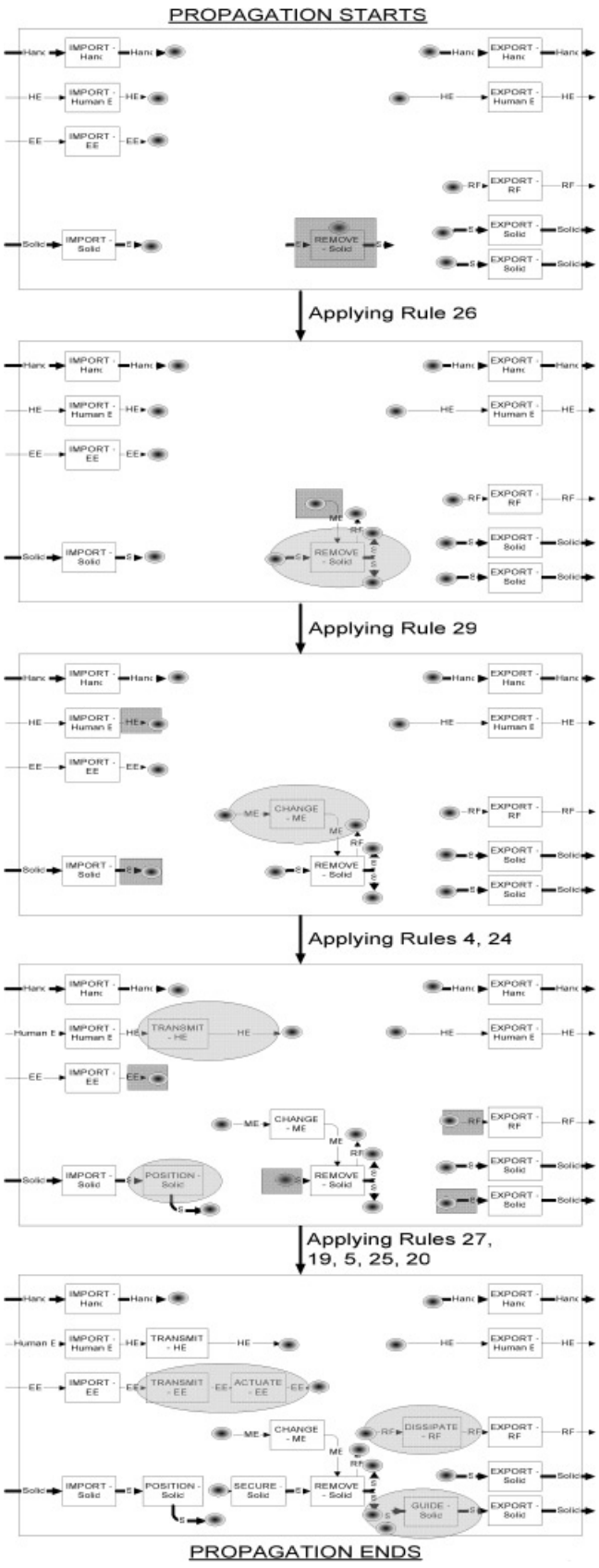


Fig. 8. A pictorial representation of the generation of the function structure in the propagation stage.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

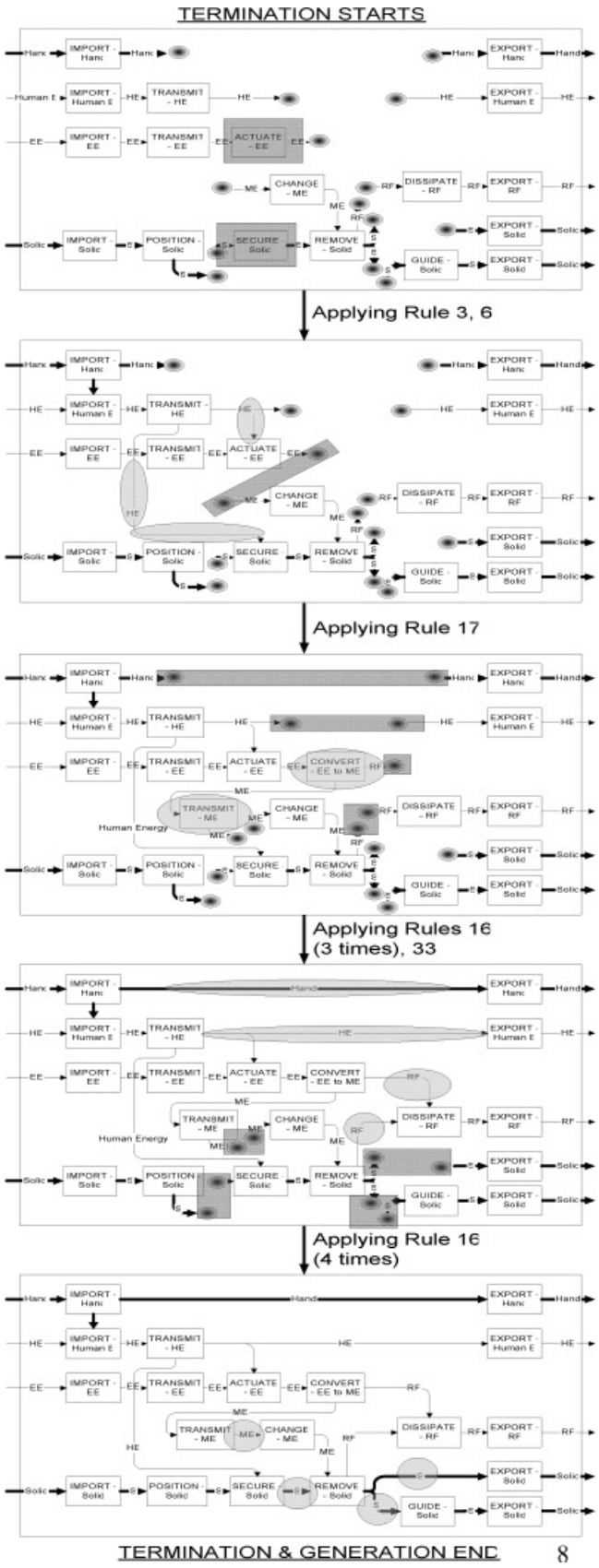


Fig. 9. A pictorial representation of the generation of the function structure in the termination stage.

only the initiation rules are active, and hence, the function structure is scanned for possible initiation rules that can be applied. It is seen that at any instant a number of rules can be applied at different active centers. Rule 1 recognizes a flow whose head is "outside" and whose tail is "inside" of the black box and adds a function Import; to that flow. Furthermore, it creates a new flow whose tail is linked to Import and head has the value "NULL" (to indicate it as an active center). Rule 2 also works in a similar way for outputs. Figure 7 shows the application of rules 1 and 2 that transforms the black box into an intermediate state with as many as nine active centers.

These nine active centers then become the input for the propagation stage (Fig. 8). Rule 26 recognizes the function Remove Solid and adds some flows in front of and at the back of that function. The application of rule 26 adds a new flow "ME," which is open at its tail. This becomes the place where the next rule, rule 29, is applied. Rule 29 adds a function Change ME to the back of the ME flow and creates another open flow at its back. Thus, we see that in the propagation stage, active centers are recognized, rules are applied, and most active centers that are removed are replaced by new ones. In the termination stage shown in Figure 9, rules act on the open and dangling flows and complete the function structure by carefully finding functions that connect between the remaining active centers. The result is the completed function structure shown at the end of Figure 9.

6. IMPLEMENTATION

Function structures can be compared to a typical graph where the vertices are represented by functions and the edges are represented by flows. It is obvious that they are directional graphs as the direction of flows is important in ensuring the validity of function structures. The data structure for a graph needs to store the vertices and edges such that each edge references to the two vertices that it connects to. Typically, graphs store information only in the vertex and the edges are used just to point from one vertex to another. In the case of function structures, we need to store information in both the vertices (functions) and edges (flows) as is discussed in Section 4. The method that we follow stores both the functions and flows in a sequential list. Each function references all edges it is associated with, and each flow stores information about the function that it is pointing from and pointing to.

Flows and functions are represented as data structures. Some of the information that is stored in a flow includes its unique number, its name, the unique numbers of the functions connected to the front and back of the flow, whether the flow is a material or an energy, and the specific type of material or energy. Similarly, functions also store relevant information like unique number, the unique number of flows connected to its front and back, information about the flow that is being processed by the function.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

Before explaining how the program works, it is necessary to elucidate the different classes that are a part of the program. There are three main classes that help in modularizing the program so that it is easy to understand.

6.1. Active center class

The active center class has methods inside it that help in creating active centers. These methods update the active centers at each stage of the process. We already saw how the generation is divided into initiation, propagation, and termination. These methods read the function structure at its current state and update the active centers in the active list. The first method *update_activelist_for_i()* updates active lists for the initiation stage, *update_activelist_for_p()* updates active centers for the propagation stage, while *update_activelist_for_t()* and *update_activelist_for_t1()* update active lists for the termination stage. The reason for having two methods for the termination stage is because termination stage occurs in two distinct steps: one where the active centers are closed, and another where the open flows are just linked to one another.

6.2. Globals class

This class has many methods that deal with the trivial, but often repeated actions of the program. The methods can be divided into four categories: one where data are processed, one where data are displayed, one where flows and functions are created, and one where the final function structure is written into a file. These methods are called from various parts of the program.

6.3. Rule application class

As the name suggests, this class contains methods wherein the rules are recognized and applied. Further, the recognition of the rules is divided into stages, and hence at each stage, only the rules that belong to that stage are recognized.

The main method in this class is the *rule_recognize()* method, which calls all the other methods. Each active center from the active list is passed to the rules of the current stage one by one. If a particular rule is recognized, then it is added onto the list of rules that can be applied. Once the active list is completely traversed, the program waits for the user to choose a rule and active center and the corresponding rule is then applied.

6.4. Description of process

The sequence of the program is as follows: scan the function structure for active centers, recognize applicable rules, create an active list, if the size of the active list is zero, then move to the next stage; if the size of the list is greater than zero, return the list of applicable rules, choose one rule (done through user interaction), and apply it. The process is repeated for the three stages because the basic framework remains the same, the only thing that differs is the set of rules that are recognized and applied.

The black box is input as a text file that has the inputs, outputs, and the primary function(s). Flows are specified starting with the kind of flow, *M* or *E*, for Material or Energy, followed by the domain of the flow (i.e., Human, Electrical, Hydraulic, etc.). A sample black box for a coffee grinder is shown in Figure 10. Currently, the terms in this text file must match the accepted set of terms used thus far. Although any translation to the reduced common basis must be performed as in the example of Figure 6, this is the only preparation needed for the implemented system to begin constructing a function structure.

6.5. Computational interface

Snapshots of the user interface at various points during program execution are shown in Figures 11, 12, and 13. Figure 11 shows a screenshot of the user interface as soon as the program is executed. There are four output text boxes in the window that display (from left to right) the applicable rule set, the active centers, the flows, and the functions of the function structure at that instant. There is also a rule set reference window at the bottom that the user can access information about any of the 69 rules. Currently, there is no graphic showing the function structure as it is being created. Although drawing such a structure is a fairly straight-

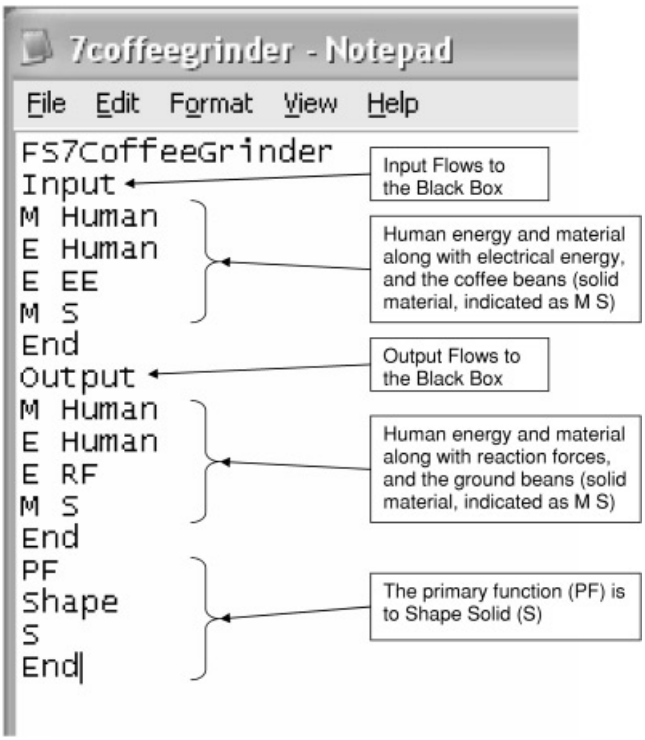


Fig. 10. A black box of a coffee grinder as it is input into the program.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

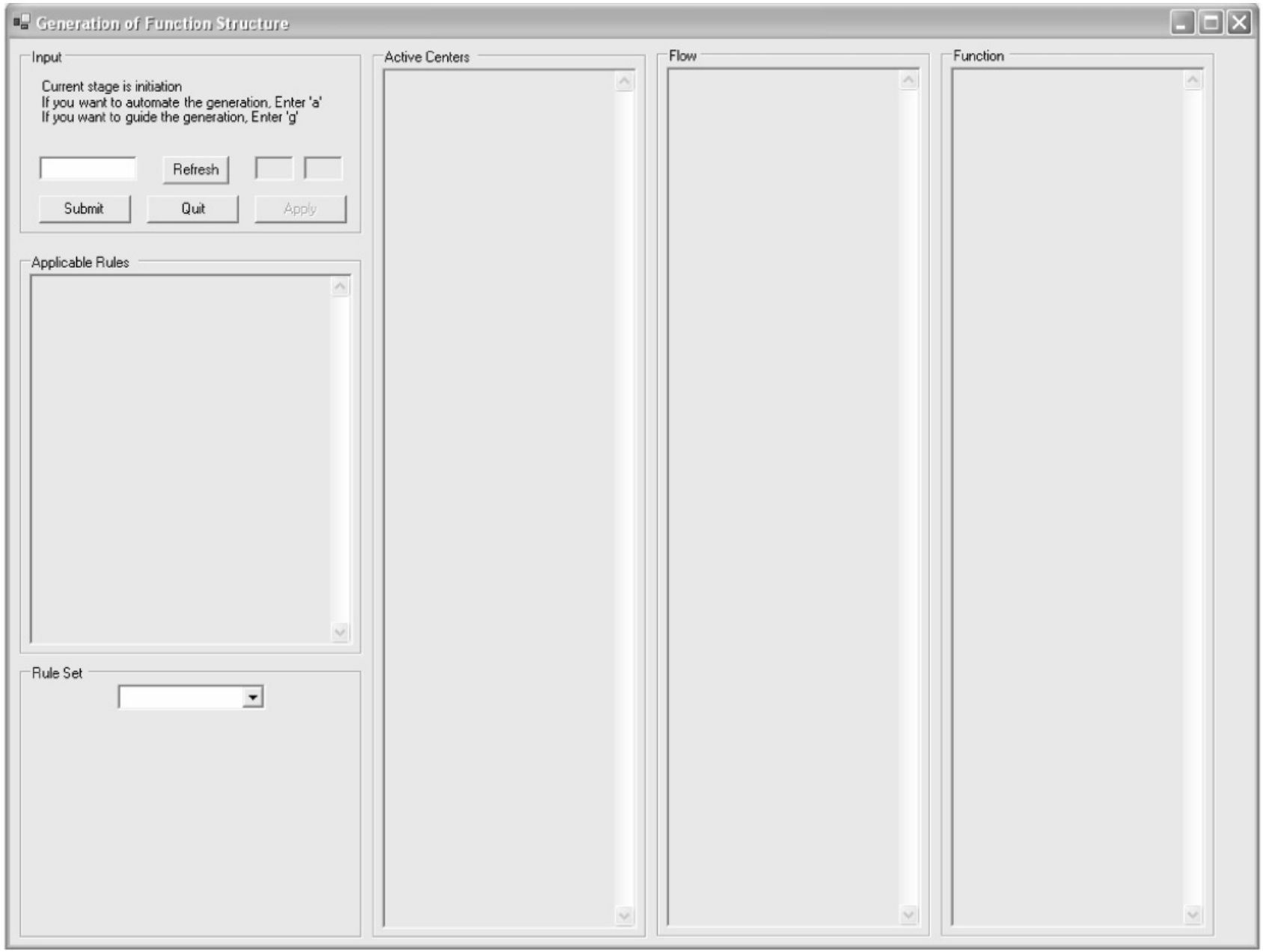


Fig. 11. A screenshot of the user interface at the onset of the process of building a function structure from a black box.

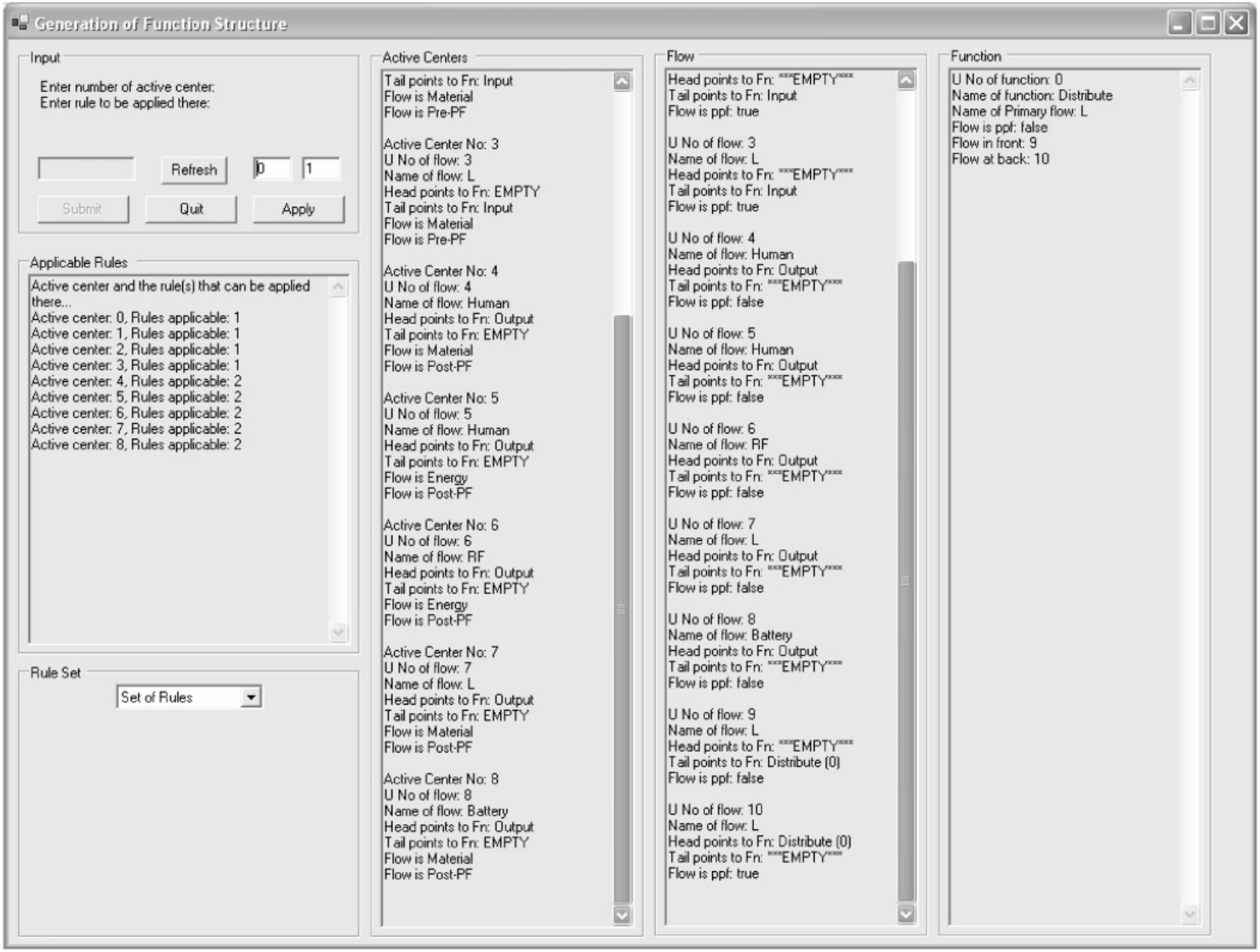


Fig. 12. A screenshot during the running of the program.

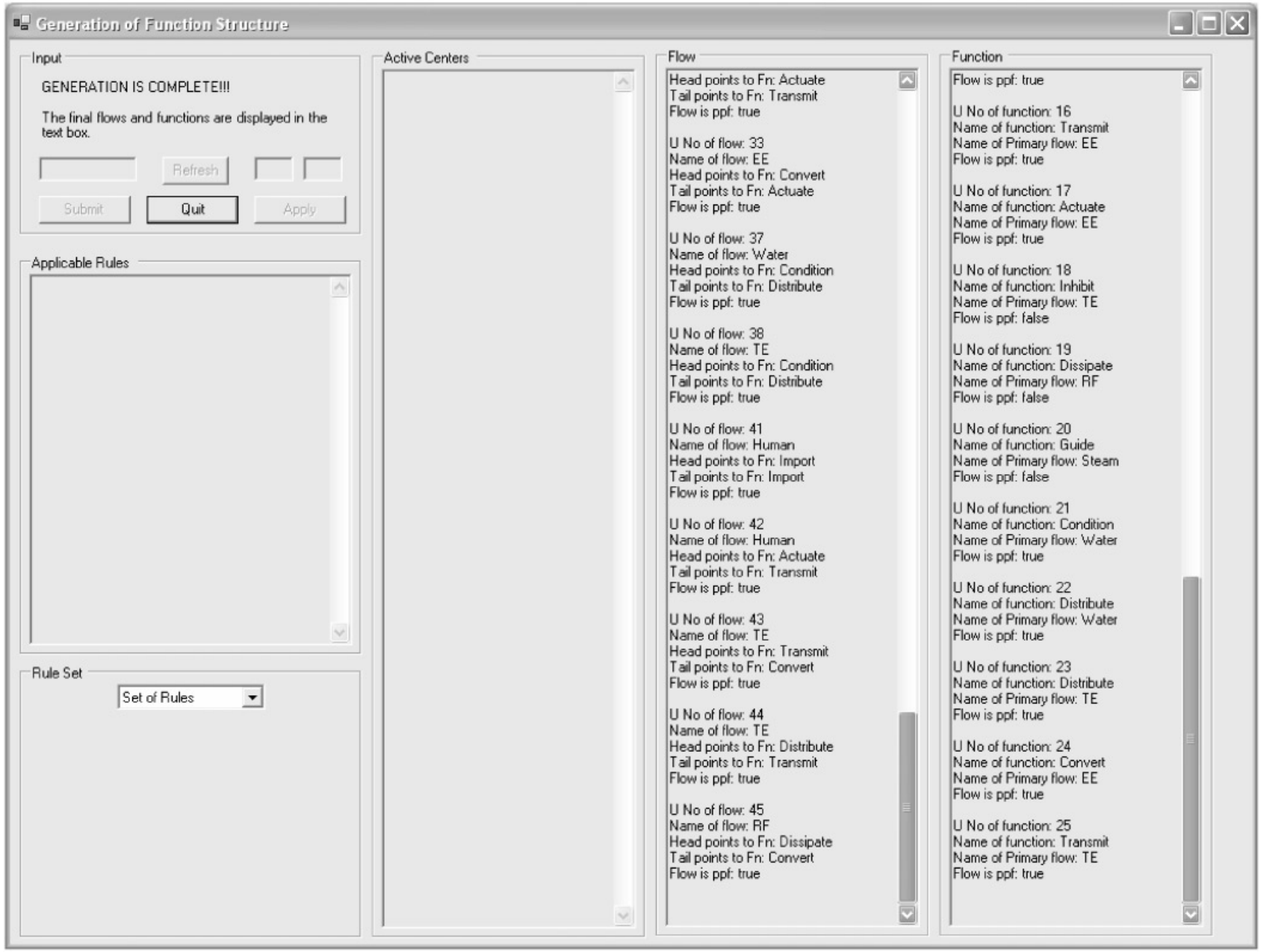


Fig. 13. A screenshot after the program is complete.

forward human activity, the heuristics followed are difficult to implement. In fact, the automatic presentation of graphs is a research area in itself. As a result, the user currently must draft the structure from the information presented in the flow and function text boxes. The user is first presented with an option to choose the kind of generation that he/she wants to pursue. If the "automate" option is chosen, then the computer will repeatedly choose rules and active centers until no more rules are applicable.

Figure 12 shows the program being executed. The "Applicable Rules" text box displays the list of active centers and the applicable rules at those active centers. The "Active Center" text box displays more information about each active center. The "Flow" and "Function" text boxes display all the flows and functions of the function structure. The user enters the number of the active center and the rule that he wants to apply at that active center. Figure 12 shows an example where there is only one rule applicable at the active centers; however, there may be a case where more than one rule is applicable at one active center. If the user has chosen the "guide" option, he/she can choose the rule that needs to be applied. This way, the user has control over the final function structure.

Figure 13 shows the program once the function structure is generated. Once the program is complete, a text file that has the function structure in a textual easy to follow format is written. The function structure of a coffee grinder is shown in Figure 14.

7. EXPERIMENTS

In this section we discuss two experiments that have been developed to test the effectiveness of the function structure grammar. First, we look at a comparison of function structures created by engineering designers using the grammar rules to function structures created by designers not using the rules (Section 7.1). Next, we look at the resulting function structures created independently by a computational process (Section 7.2).

7.1. Effectiveness of grammar rules applied by hand

To check whether these rules provide a framework to designers that actually helps create better function structures, an experiment is conducted. A set of seven graduate students was divided into two groups of three and four. These students had recently studied in a design methodology course that taught how to create high-quality function structures. Each of the groups of students was given a product for which they had to draw the function structure. The products were then exchanged in the groups and function structures were again created, this time using the grammar rules as a guideline.

These function structures were then graded for quality using a double-blind system. The grader (another graduate student who had been a teaching assistant for a design meth-

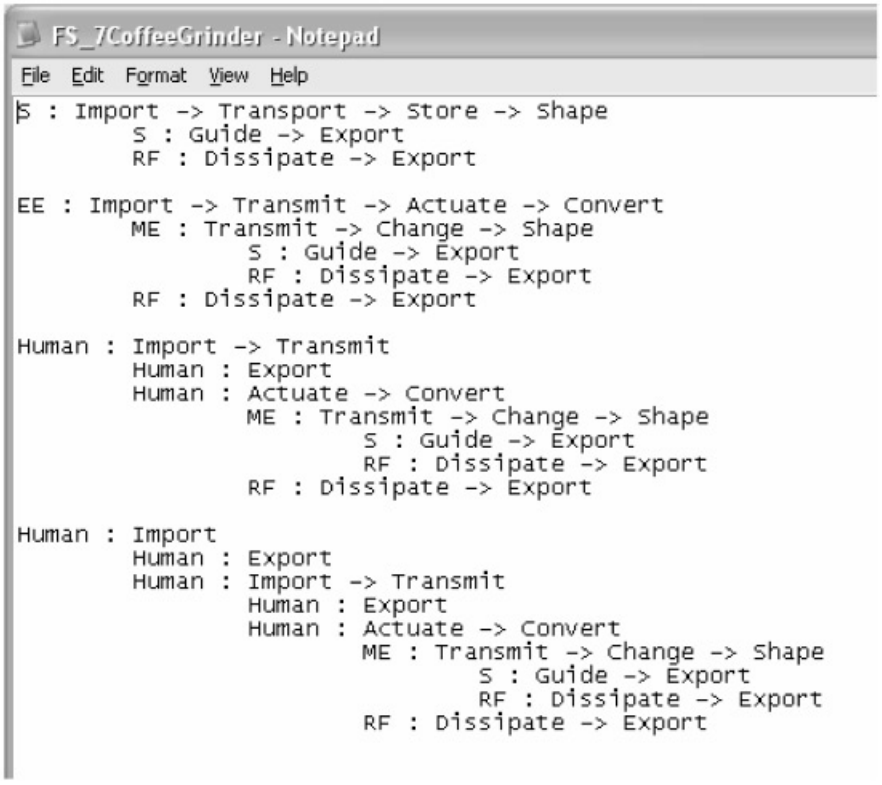


Fig. 14. A completed function structure of a coffee grinder returned at the end of the process.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

odology class) was given a shuffled stack of function structures and was not aware of the experimental details, or the existence of the grammar. The strategy that the grader used to determine a quantitative score for the quality of a function structure was done by averaging separate scores for consistency, level of detail, completeness, and creativity. It is not clear how the grader defined such terms, but when questioned, the grader indicated that he initially determined what he thought the best function structure for a given product should be and then compared each specimen to his ideal. He said he was looking at how well the students understood the basic physics of the product and how thorough they were at identifying all functions that would represent significant and subsequent design efforts.

The results are shown in Figure 15. It is easily observed that the function structures that were created using the rules score better than the ones done without the function structure grammar. As the figures indicate, nearly all of the highest scoring function structures were done with the aid of the grammar. In fact, students who scored poorly when the gram-

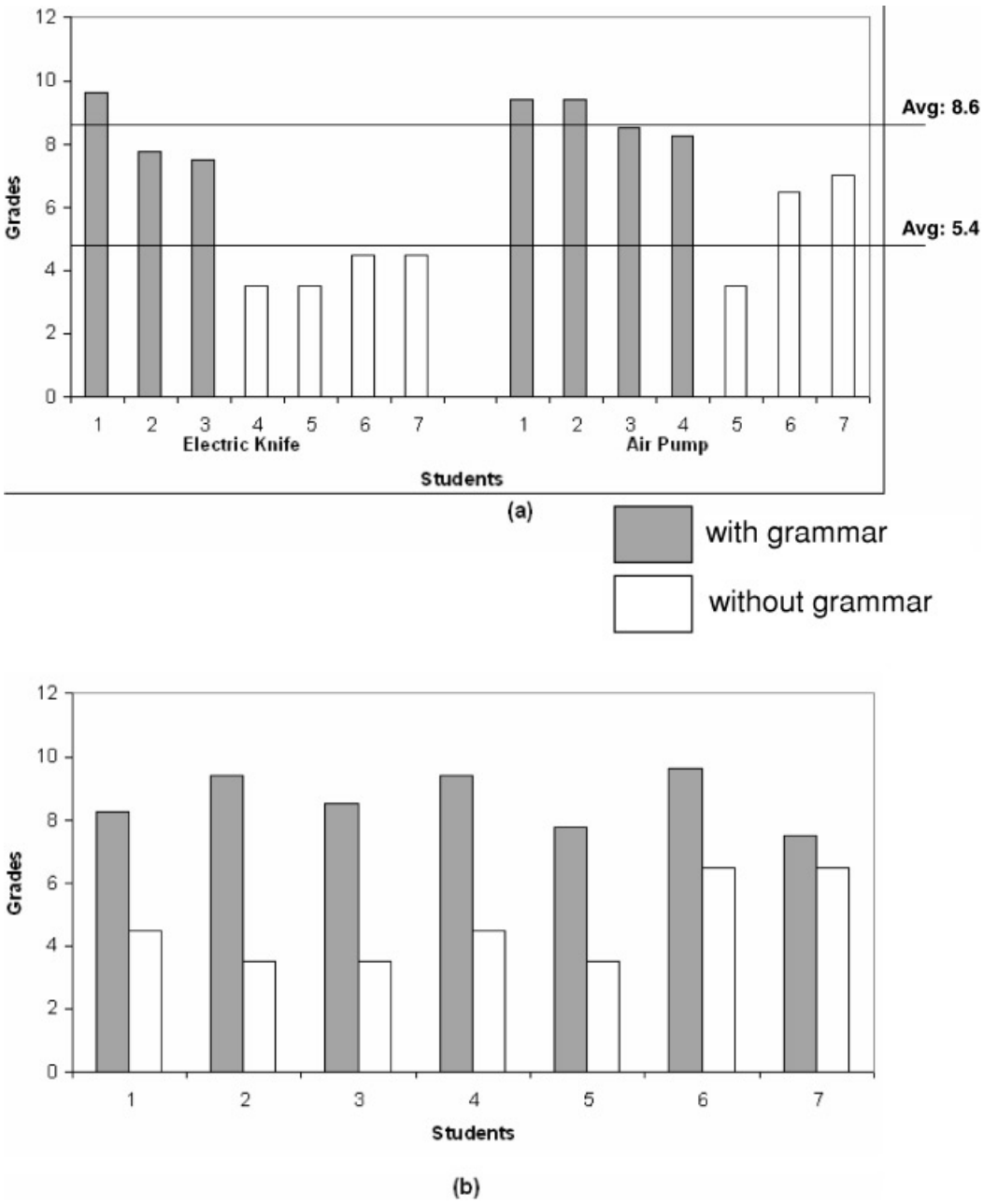


Fig. 15. A graph between grades and students. (a) The grades obtained using the rules are the dark bars (average = 8.48) and the grades obtained without rules are the light bars (average = 4.8). (b) A graph showing the performance of the students using rules and not using rules.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

mar was not used improved greatly when aided by the grammar (see Fig. 15b). A *t* test was conducted comparing students' scores when the grammar is used to scores when the grammar is not used. The mean score for structures made with the rules is 8.63, with a standard deviation of 0.85. This is shown to be significantly higher (a *t* test of seven samples yields a value of 3.41 or a probability of less than 0.05 that the null hypothesis is true) than the mean score for those created without the rules (mean = 5.38, standard deviation = 2.21).

After the grading, we questioned the grader for his general thoughts about the function structures. He mentioned that there seemed to be two sets of students: those that put the time into creating quality function structures, and those who did not understand all of the concepts. When we showed the grader the two function structures created by student #2 (one with a score of 3.5 and one with a score of 9.375), the grader was surprised, and conjectured that the student must have run out of time in creating the electric knife example. In fact, student #2 had created the knife function structure first on his own and the air pump structure using the grammar rules. When asked to order the function structures that would most aid in the design of one of these products, the grader ranked three of the function structures created by the grammar as the best. Of the poorest function structures, the grader said that these students did not spend enough time to capture the subtleties of the products and that performing the exercise would not help subsequent design activities. He hoped that such students were using an alternate method (to creating function structures) to properly explore the conceptual design phase.

7.2. Automated development of function structures

As described in Section 6, the user can choose to have the computer automate the search for a complete function structure. Because the grammar rules encapsulate the feasible additions that can be made, the automation is based simply on choosing a series of applicable rules. In this example, we use the black box for the palm sander as shown in Figure 16. Currently, we are not able to systematically search the entire tree. Instead, we simply run through 20 controlled searches through the tree to compare whether the created solutions are valid. Interestingly enough, of the 20 searches only three unique solutions were created (see Fig. 17). Fifteen of the 20 were identical (solution A), and only 2 more were found in the remaining 5 searches (see Table 4). All three solutions appear to follow the rules of creating a logical function structure and provide a clear basis of the functional requirements needed in designing a palm sander.

As is described in Table 4, the three solutions are similar in many aspects; however, there are a few small differences. Solution B is the same as A with the exception of one function, "guide air." In the black box, air is specified as an input and output along with pneumatic energy to charge the air on output. This forces the design to connect the air flow through a series of functions so that it can be expelled to blow away sawdust. In solution B, the air is simply "exported" after it is "transported" (where the transport function requires the participation of an energy flow, such as rotational mechanical energy, i.e., a fan). One can imagine that if embodied designs were created for solutions A and B, they would differ only in the fact that A would have some channel to "guide" the air from the fan to the outlet on the product housing where it is exported, whereas B would lack this channel and simple have the fan placed at the exit of the air on the housing. Solution C is interesting in that it divides the electrical energy into two flows once it is "imported" and uses these flows to drive the sander and transport the air separately. This may be desirable so that the fan can be turned on and off separately from the sander.

8. DISCUSSION

With an average of 20 functions per function structure and 10 rules applicable at any stage, there would appear to be 10^{20} possible solutions. However, after 20 different searches through the tree, only 3 unique solutions are found. We believe there are many repeated states and the tree is much smaller than this. This is most likely due to the fact that many rules do not negate the application of another, because many of the rules look at single active centers independently of other active centers. For example in initiation, we simply apply rules which introduce "importing" and "exporting" of the various flows crossing the black-box boundary. Once these are completed, we predict that only a single state is possible at the beginning of the propagation stage, as is shown in Figure 18. We believe that the propagation and termination stages also contract before completion, thus

Table 4. *The summary of the three function structures created in the automated search (A, B, and C)*

Unique Function Structure	Found Form Search #	Defining Characteristic	Valid
A	1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 18, 20	Output air is *guided* before being exported, and fan and sander are driven by same electrical energy path.	Yes
B	8, 9, 16	Same as A but air is simply exported	Yes
C	17, 19	Same as A but electrical energy is divided immediately after importing into two circuits: one to spin the fan, and one to actuate the sander	Yes

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

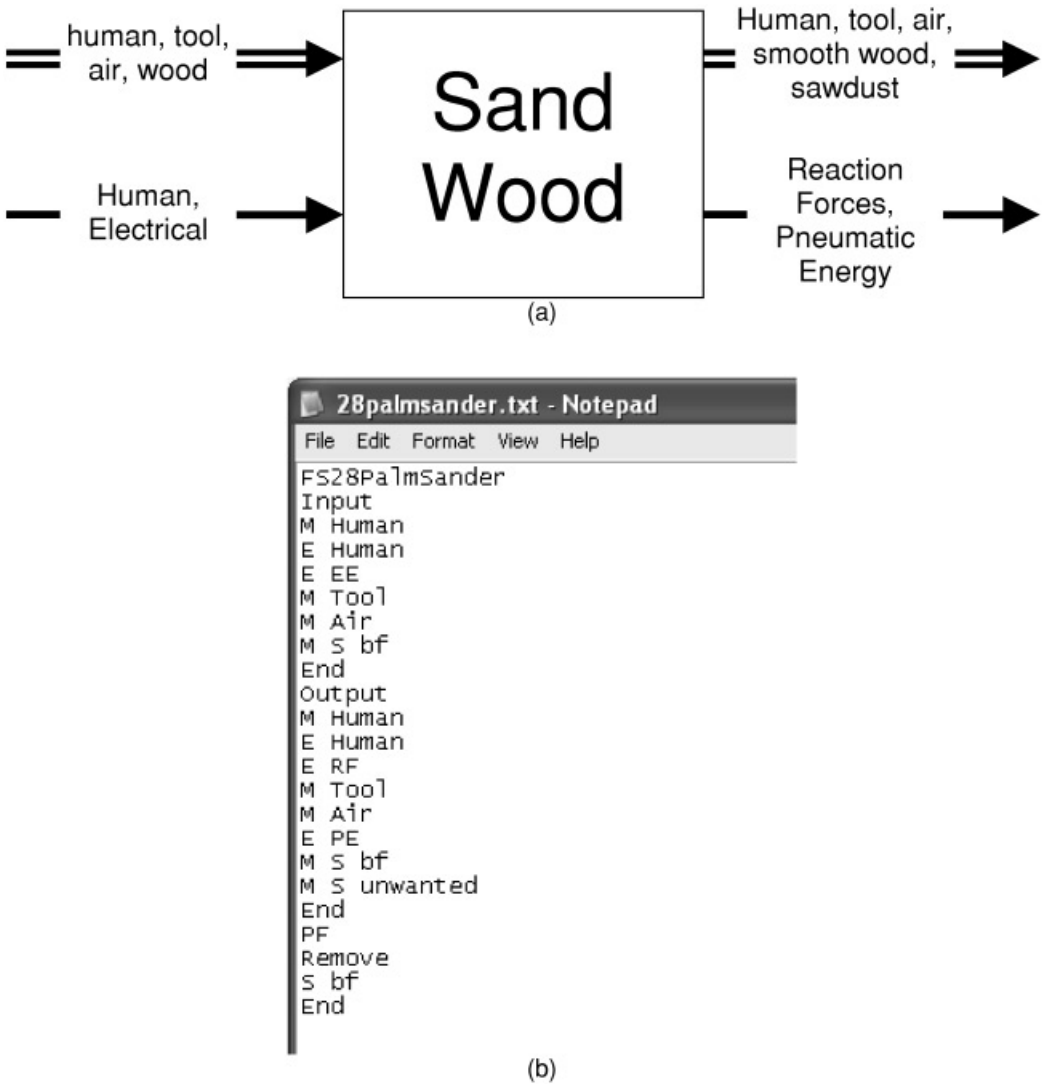


Fig. 16. (a) A black box for a palm sander and (b) the reduced textual representation of the black-box input into the program.

resulting in only a small number of solutions from a potentially large search tree.

One may be dubious as to why so few valid function structures exist. This results from the fact that the black boxes in our examples are very detailed due to the formulation of the grammar rules. For example, air flow and pneumatic energy is prescribed in the palm sander problem, although it is not crucial to the primary function of the palm sander. Because rules are constructed to connect all input and output flows we are, in a way, constrained mostly by this initial black box. If one were to create a set of grammar rules for a different domain of artifacts, or for a different formulation of function structures, it is not clear that the search tree will reduce as dramatically. In general, the search through the tree of possible function structures may need to be tackled by a more comprehensive search strategy where, for example, intermediate solutions are evaluated and a computational decision maker chooses rules based on past performance or through some stochastic search mechanism.

In the products that were examined, it was observed that, where heat energy was not wanted, the thermal energy was inhibited before being exported. However, in the case of an electric iron, thermal energy is a desired output. Hence, a new parameter is added to flows that classified the flows as *wanted* or *unwanted*. In the case of a flow being unwanted (e.g., thermal energy in many cases and dirt in a vacuum cleaner) then it is specified in the black-box model. For example, in Figure 16b, we see that although sawdust is not distinctly used to characterize the output material, this material is specifically listed as "unwanted." Different grammar rules are valid for wanted and unwanted flows. Additionally, the concept of *boundary flows* is also developed. It is observed in many products that there are flows, usually the flow going into the primary function, which interacts with the physical boundary of the products. Examples are leaves in leaf blower and hair in a hair dryer. These flows always follow specific rules that are different from other typical flows. Other classifications like the above were made for

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

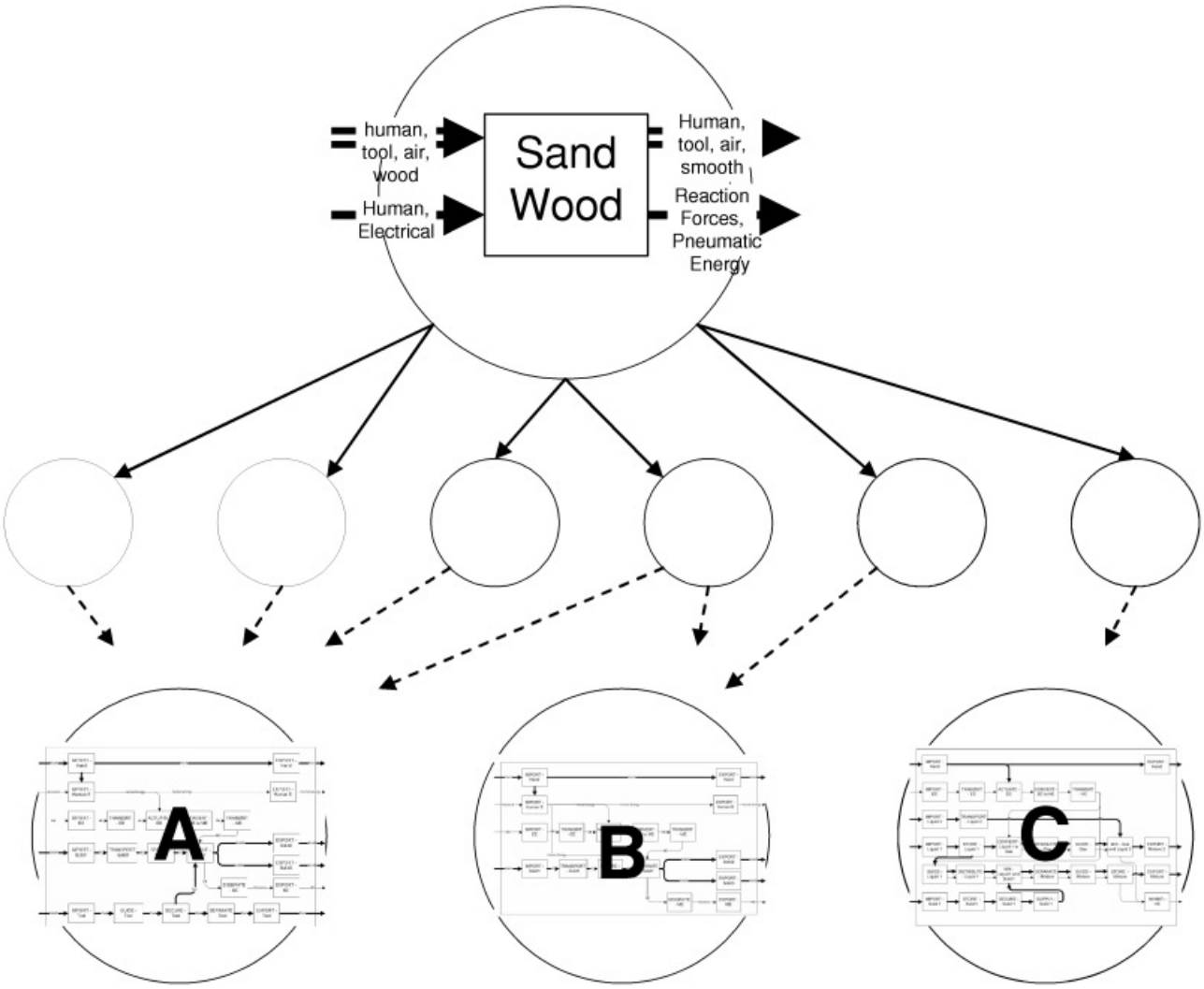


Fig. 17. Of the 20 searches through the tree, only three unique function structures were created for the palm sander.

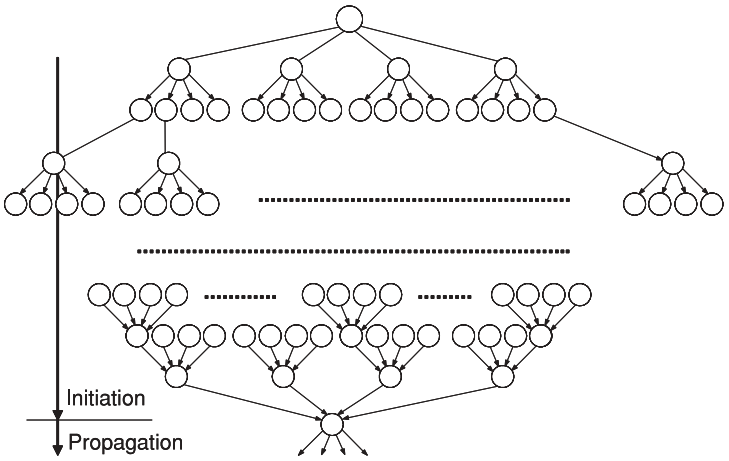


Fig. 18. Despite the many possible paths that can be followed in the tree, it is believed that only one or a few states exist at the end of each stage (initiation, propagation, and termination).

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

specific flows such as water and steam. Although the flow "liquid" encompasses water, and "gas" encompasses steam, classifications like this make the rule recognition more compact and efficient. Furthermore, flow classifications like "Tool," "Battery," and "Container" were also added because of their specific behavior.

Making the rules depend on the initial flows crossing into and out of the black box and further qualifying flows as wanted, unwanted, and/or boundary has lead to a structured set of required functions and thus only a few valid function structures. Although the original motivation of the research is to enable the generation of multiple solutions, this result is perhaps more informative and useful. Perhaps a future implementation could suggest new flows to the black box or ignore prescribed flows in hopes of deriving new solutions.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a graph grammar, where a set of rules has been developed based on the function structures of 30 products. The rules that were formed by observing trends in the function structures initially created by hand and by general principles like using gears and hydraulics common to so many products. Experiments were conducted on the grammar by comparing a set of function structures generated by students applying the grammar to a set developed by students who were not given the grammar. The experiment shows that more consistent results are created by students using the grammar, and it does not seem to hinder the creativity of subsequent generated concepts thus proving that the rule set is useful.

One challenge that is prevalent with many grammars is that they are difficult to implement. However, our approach to examining many products and adopting the active center concept from polymerization has helped us to create a fully implemented Scenario 5 grammar (see Fig. 3). The rules are based on adding new functions and flows to active centers in the function structure until there are no more active centers.

Our main focus is to create a formalism that would help in realizing the full potential of function structures. This contribution could make function structures better accepted and widely used. The desire to automate the generation of function structures can in no way be considered as a replacement for the thought process that one undergoes while creating function structures. It is our belief that these rules would help people associated with design methodologies to accept and teach function structures more efficiently as well as to brainstorm new functional strategies that have not been considered by the user.

Future work is aimed at computationally searching the entire tree of solutions to determine how much variability exists in creating a function structure from a black box, and in expanding the grammar to include more rules by examining more products outside the scope of those shown in Table 1.

REFERENCES

Cagan, J. (2001). Engineering shape grammars. In *Formal Engineering Design Synthesis* (Antonsson, E.K., & Cagan, J., Eds.). New York: Cambridge University Press.

Cagdas, G. (1996). A shape grammar: The language of traditional Turkish houses. *Environment and Planning B: Planning and Design 23(4)*, 443–464.

Chandrasekaran, B., Goel, A. K., & Iwasaki, Y. (1993). Functional representation as design rationale. *Computer 26,* 48–56.

Chase, S.C. (1998). User interaction models for grammar based design systems. *International Journal of Design Computing 1*, Paper 2. Available online at www.arch.usyd.edu.au/kcdc/journal/vol1/dcnet/stream4/paper2/

Chen, L., Jayaram, M., & Xi, J.F. (2002). A new functional representation scheme for conceptual modeling of mechatronic systems. *Proc. ASME 2002 Design Engineering Technical Conf.,* Paper No. DETC2002/DTM-34012, Montreal, Canada.

Collins, J., Hagan, B., & Bratt, H. (1976). The failure-experience matrix—a useful design tool. *Journal of Engineering for Industry 98*, 1074–1079.

Fu, Z., De Pennington, A., & Saia, A. (1993). A graph grammar approach to feature representation and transformation. *International Journal of Computer Integrated Manufacturing 6(102)*, 137–151.

Gero, J.S., & Kannengiesser, U. (2003). The situated function–behaviour–structure framework. *Design Studies 25(4)*, 373–391.

Gietka, P., Verma, M., & Wood, W. (2002). Functional modeling, reverse engineering, and design reuse. *Proc. ASME 2002 Design Engineering Technical Conf.,* Paper No. DETC2002/DTM-34019, Montreal, Canada.

Hirtz, J., Stone, R., McAdams, D., Szykman, S., & Wood, K. (2001). Evolving a functional basis for engineering design. *Proc. ASME 2001 Design Theory and Methodology Conf.,* Paper No. DETC2001/DTM-21688, Pittsburgh, PA.

Hubka, V., Andreasen, M., & Eder, W. (1988). *Practical Studies in Systematic Design*. London: Butterworths.

IEEE Computer Society. (1998). *IEEE Standard for Application and Management of the Systems Engineering Process,* pp. 1220–1998. New York: IEEE.

Kirschman, C., & Fadel, G. (1998). Classifying functions for mechanical design. *Journal of Mechanical Design 120*, 475–482.

Klahr, D., Langley, P., & Neches, R., Eds. (1987). *Production System Models of Learning and Development*. Cambridge, MA: MIT Press.

Koning, H., & Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B: Planning and Design 8*, 295–323.

Kurfman, M., Rajan, J., Stone, R., & Wood, K. (2001). Functional modeling experimental studies. *Proc. ASME 2001 Design Engineering Technical Conf.*, Paper No. DETC2001/DTM21709, Pittsburgh, PA.

Li, X., Schmidt, L., He, W., Li, L., & Qian, Y. (2001). Transformation of an EGT grammar: new grammar, new designs. *Proc. ASME 2001 Design Engineering Technical Conf*., Paper No. DETC2001/DTM-21716, Pittsburgh, PA.

Longenecker, S.N., & Fitzhorn, P.A. (1991). A shape grammar for non-manifold modeling. *Research in Engineering Design 2(3)*, 159–170.

Otto, K., & Wood, K. (2001). *Product Design: Techniques in Reverse Engineering and New Product Development*. Upper Saddle River, NJ: Prentice–Hall.

Pahl, G., & Beitz, W. (1984). *Engineering Design: A Systematic Approach*. London: Design Council.

Pinilla, J.M., Finger, S., & Prinz, F.B. (1989). Shape feature description using an augmented topology graph grammar. *Proc. NSF Engineering Design Research Conf*., pp. 285–300, Amherst, MA, June 11–14.

Progelhof, R.C. (1993). Polymer engineering principles: Properties and tests for design. In *Polymer Science* (Gowariker, V.R., Viswanathan, N.V., & Sreedhar, J., Eds.). New York: Wiley.

Schmidt, L.C. (1995). *An implementation using grammars of an abstraction-based model of mechanical design for design optimization and design space characterization*. PhD Thesis. Carnegie Mellon University.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design 7*, 343–351.

https://doi.org/10.1017/S0890060405050110 Published online by Cambridge University Press

Stone, R., & Wood, K. (2000). Development of a functional basis for design. *Journal Mechanical Design 122*, 359–370.
Szykman, S., Bochenek, C., Racz, J.W., Senfaute, J., & Sriram, R.D. (2000). Design repositories: Next-generation engineering design databases. *IEEE Intelligent Systems 15(3)*, 48–55.
Ulrich, K.T., & Eppinger, S. (1995). *Product Design and Development*. New York: McGraw–Hill.
Wang, K., & Yan, J. (2002). An analytical approach to functional design. *Proc. ASME 2002 Design Engineering Technical Conf.,* Paper No. DETC2002/DTM-34084, Montreal, Canada.

Prasanna Sridharan obtained his MS in mechanical engineering from the University of Texas at Austin in December 2003. His research interests are function structures and enhancing design methodology by automating the design process. He is currently working for a software company in Delaware.

Matthew Campbell received his PhD from Carnegie Mellon University in the summer of 2000. He has since been an Assistant Professor at the University of Texas in the Mechanical Engineering Department. Dr. Campbell is a member of the AAAI, the AIAA, Phi Kappa Phi Honor Society, the ASME, the ASEE, and the Design Society. He has been acknowledged with best paper awards at conferences sponsored by the latter three. His research focuses on computational methods that aid the engineering designer earlier in the design process than traditional optimization would. He was the recipient of the CAREER award for research into a generic graph topology optimization method. This research represents a culmination of past computational synthesis research including the automatic design of sheet metal components, multistable MEMS devices, MEMS resonators, function structures, and electromechanical configurations.