Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2002), *16*, 273–290. Printed in the USA.
Copyright © 2002 Cambridge University Press 0890-0604/02 $12.50
DOI: 10.1017/S089006040216402X

Designing inner hood panels through a shape grammar based framework

JAY P. MCCORMACK AND JONATHAN CAGAN
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213, USA

(RECEIVED August 24, 2001; ACCEPTED May 15, 2002)

Abstract

A framework for a design tool based on shape grammars is presented as an effective means for supporting the early stages of design. The framework uses a shape grammar interpreter to implement parametric shape grammars, allowing the grammar to be used interactively by a designer or optimization routine. A shape grammar to design inner hood panels of vehicles is introduced as an example of a parametric engineering shape grammar, and it is used with the framework to create standard and novel designs made possible by rules that take advantage of shape emergence.

Keywords: Design Tool; Engineering; Shape Grammars

1. INTRODUCTION

Many classes of products and devices undergo repeated redesign because of changing design parameters and constraints. For these types of design problems, it is useful for the engineers and designers involved to have a computational tool that can build the product or part based on experience. Ideally, that tool would possess three key features: it would have a way to store knowledge about the class of products and the logic to successfully generate one of the products through interactions with a human user or computer process; it would reason directly with a sufficiently detailed geometric representation of the product, and the level of detail necessary would be dependent on the geometric intricacies of the product involved; and the tool would support the creation of novel instances of the product. Tools exist in the form of expert systems, which can perform some of the desirable tasks but with significant limitations. Traditional expert systems lack a sufficient geometric representation and general approach to reasoning about product forms. Additionally, they are unable to produce innovative designs that are more than just the sum of different combinations of the information contained within its set of rules.

In contrast, shape grammars have recently been shown to be an effective engineering design tool. They succeed in areas where traditional expert systems have failed because of their direct handling of and reasoning about geometry, their ability to operate on a parametric geometric representation, and their ability to support the emergence of shape (Agarwal & Cagan, 2000). A shape grammar defines the language of a product or part through a set of rules, which contain the logic behind the design of the device. Shape grammars are able to create novel designs through the emergence of shapes that are not directly added to the design but rather are created through the algebraic properties of shape.

This paper introduces a parametric shape grammar that models the inner hood panel of automobiles. The inner hood panel is the structural panel that is joined with the visible outer hood panel to form the hood assembly. The part consists of structural beams, hinges, the striker, hood support means such as a prop rod or gas struts, and other hood operational features. The properties of the hood, such as frequent redesigns and geometric features that define its function, make the inner hood an appropriate product to be modeled with a shape grammar. While providing the rigidity to the hood assembly, the inner hood panel must be designed to fit around the engine, battery box, and shock towers. Redesign of the inner hood panel is also necessary if any one of a number of related components is altered such as modifications to the outer hood panel appearance or

Reprint requests to: Jonathan Cagan, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA. E-mail: cagan@cmu.edu.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

changes in size or location of the uppermost components in the engine compartment. The inner hood panel shape grammar is the first engineering shape grammar to take advantage of emergence, which makes possible the generation of novel items, in this case, hoods. Figure 1 shows an actual inner hood, a representation of the structure of the same hood from the shape grammar (with labels removed for clarity), and a novel hood that was generated by an optimization routine.

This paper also presents a framework for a CAD system based on shape grammars, as shown in Figure 2. This framework allows us to illustrate the power of the hood grammar and its feature of emergence, but it also provides a means to implement and use shape grammar technologies in design practice. The rules of the shape grammar are implemented with a shape grammar interpreter. A user working with the grammar through an interactive loop chooses which of the applicable rules to apply and where to apply it. Alternatively, an optimization routine can make the choices presented by the interpreter concerning rule selection, application location, and parameters during instantiation. In this application an agent-based optimization technique associates the shape grammar rules with individual agents and provides an effective means for configuration optimization. After each rule application, the design is examined and, if it is deemed completed, it is passed along to the user; if not, the design is again modified by the user or the optimization routine. Unlike other engineering shape grammars, the hood grammar uses shape matching for determining rule applicability and location. Therefore, the interpreter has to be able to perform parametric shape recognition to implement the inner hood shape grammar and remain flexible in order to implement future grammars. The interpreter and the optimization routine are each described and demonstrated through their interactions with the shape grammar describing inner hood panels.

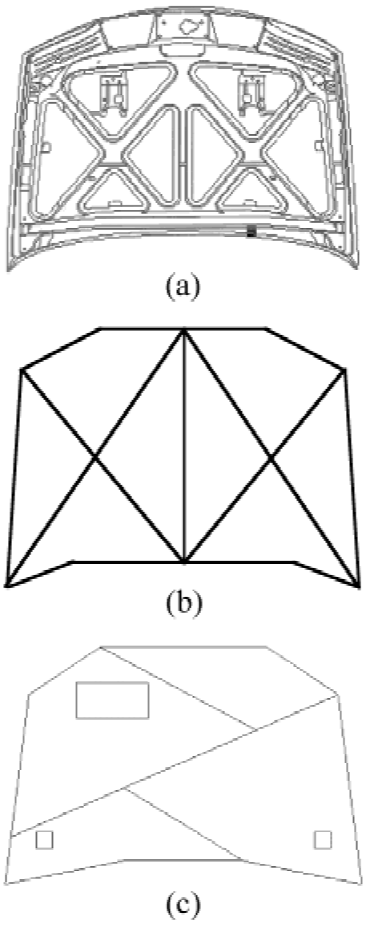


Fig. 1. (a) A real inner hood panel, (b) our representation of the real hood beam geometry with the labels not shown, and (c) a novel hood generated automatically.

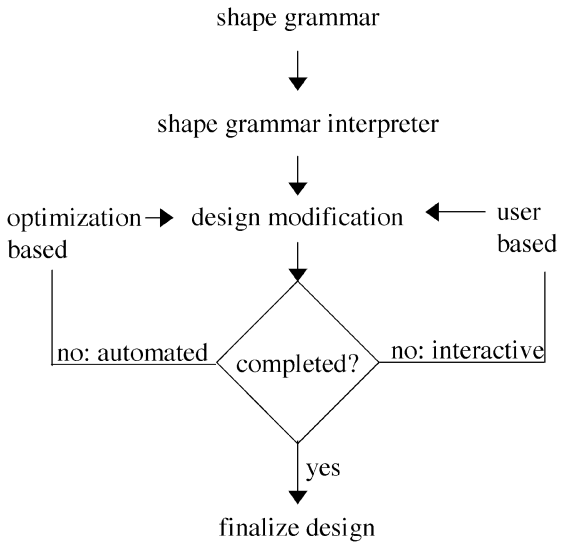


Fig. 2. The framework of the shape grammar based design tool.

2. SHAPE GRAMMAR BACKGROUND

A shape grammar (Stiny, 1980) is a set of rules based on shape that is used to generate designs through a series of rule applications beginning with an initial shape. Rules take the form of $a \rightarrow b$, where a and b are both shapes. A rule is applicable if the shape on the left-hand side, a, can be found in design shape c by applying a set of transformations to shape a. One such rule and design shape are shown in Figure 3. If the rule is applied, the shape on the left-hand side

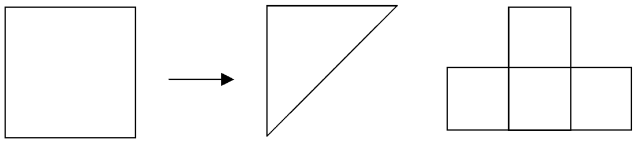


Fig. 3. A rule composed of shapes a and b and a design shape c.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

is subtracted from the design and that on the right-hand side is added to the design $c - \tau(a) + \tau(b)$, where shapes a and b undergo a transformation τ according to the transformation required to make shape a a subshape of shape c. Figure 4 shows one potential shape match (highlighted in bold) for shape a in shape c from Figure 3. Application of the rule in Figure 3 to the shape match in shape c can produce one of the four shapes on the right side of Figure 4, depending on orientation of the matched rectangle.

Shape grammars, which have their roots in the architectural literature (Stiny & Gips, 1972; Stiny, 1980), have recently found application in engineering. For example, Agarwal and Cagan (1998) introduced the coffeemaker grammar [Fig. 5(a)], Brown et al. (1994), the lathe grammar, Shea and Cagan (1997), the truss grammar, Agarwal et al. (2000), the MEMS resonator grammar [Fig. 5(b)], and Pugliese and Cagan (2001), the Harley motorcycle grammar. For further reference and discussion on engineering shape grammars, see Cagan (2001).

3. INNER HOOD PANEL SHAPE GRAMMAR

The inner hood panel designer is presented with various hood parameters concerning the placement and shape of the inner hood. The designer then follows a logical pattern to create the hood, according to the methods learned from previous designers and through experience. The designer is first given the shape of the outer hood panel and front fender and the location and shape of the automobile's frame in the vicinity of the hood. From this information the designer uses a rule of thumb to determine where the side cut lines of the hood are located. The location of the windshield glass then dictates the placement of the hood cut line near the windshield, and the front cut line is established last, based on the side cut lines and the frame structure in the front of the automobile. By following this process, the designer establishes the outline of the inner hood panel. From here the designer places the operational features and the structural beams, which avoid obstacles using structural integrity criteria as a guide. This logic behind the design of the inner hood is captured in the shape grammar rules.

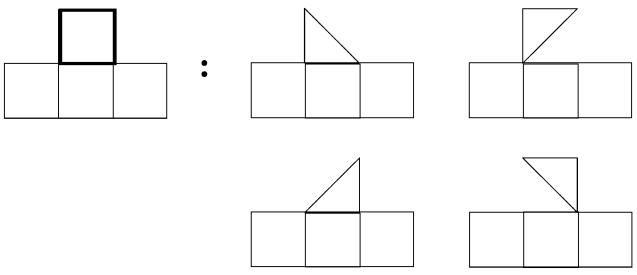


Fig. 4. A shape match of shape a in shape c is shown in bold. The four shapes that can be produced from rule application to the highlighted matching shape are also shown.

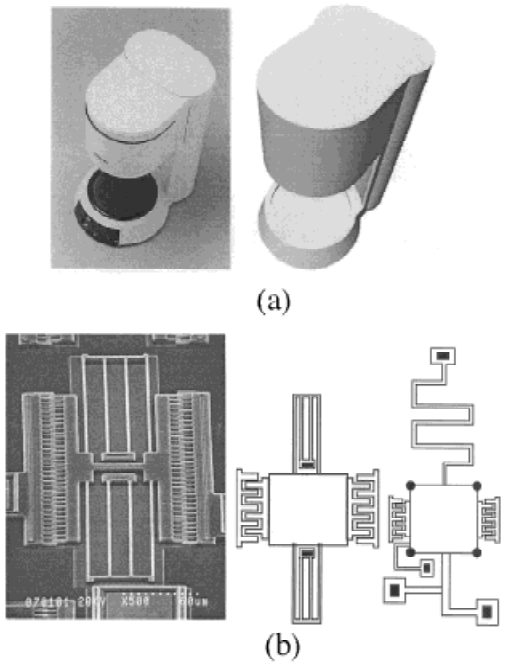


Fig. 5. (a) Actual and grammar generated coffeemakers and (b) actual and grammar generated MEMS resonators.

The inner hood panel was modeled with a shape grammar consisting of 28 rules that describe the addition and modification of structural members and operational features. The grammar was then implemented with our parametric shape grammar interpreter (McCormack & Cagan, 2000, 2001), and the process was ultimately linked with an agent-based optimization approach. The shape grammar interpreter enabled the implementation of the emergent properties of the hood grammar, which allowed powerful, encompassing rules to be written and novel designs to be created.

The inner hood panel shape grammar is a 2-dimensional grammar whose shapes are defined using points, lines, and labeled points and lines. The initial shape (Fig. 6) consists of the beams that make up the outer portion of the structure and rectangles, which represent the packaging obstacles around which the structural beams must be placed. Lines labeled b represent beams and lines labeled o represent obstacles. In order to represent a 3-dimensional object with a single line, labels are used to store additional information. The label t stores the thickness of the beam, and the label w stores the beam width. Label h on beams and obstacles stores the height of the beam or obstacle from a datum (Fig. 7). By comparing the numeric values of the h labels of an obstacle and a beam, the designer can determine if a beam passes over an obstacle or if the obstacle and beam occupy the same space. The list of labels to the side of the initial shape represents the global labels. The global labels are not associated with any geometric features but must also be matched when determining rule applicability. The

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

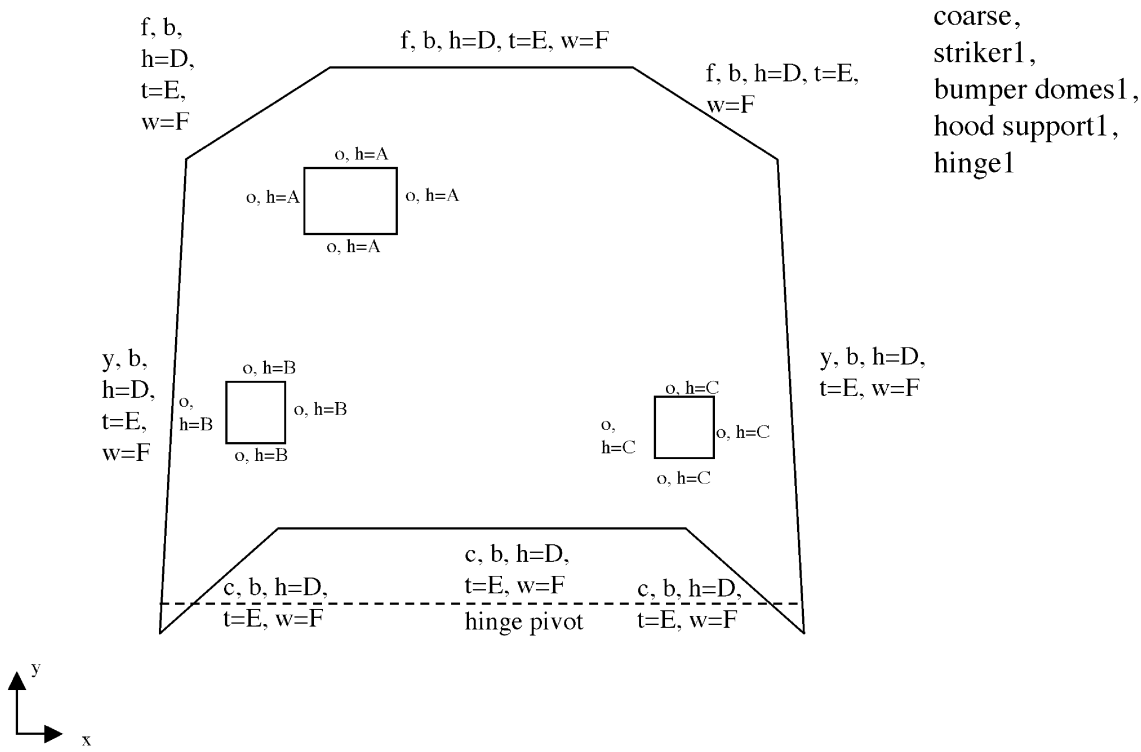


Fig. 6. The initial shape.

line in the initial shape labeled *hinge pivot* marks the location where the hinges attach to the inner hood. The global labels *striker1*, *bumper domes1*, *hood support1*, and *hinges1* found on the initial shape are used to ensure the placement of the proper number of their associated features.

The rules are divided into two sets using the global labels *coarse* and *fine*. These labels refer to the level of detail within the rule. By dividing the design process into two stages, shape searches can be performed more quickly because the coarse representation contains fewer geometric entities than the fine representation. During the coarse setting the designer can add, subtract, and move geometric and operational features of the hood. The width of a beam is represented with labels during the coarse stage, allowing the rules that operate on instances of obstacle interference to modify shapes more easily because there are fewer line segments in the representation. The fine setting allows the designer to check for minor interference by representing the width of beams geometrically, but the designer cannot adjust most features.

Rules 1–7 in Figure 8 deal with the creation and placement of hood components and obstacles in the coarse stage of design. Application of rule 1 adds a beam between two other beams. Rule 2 adds a point that marks the location of

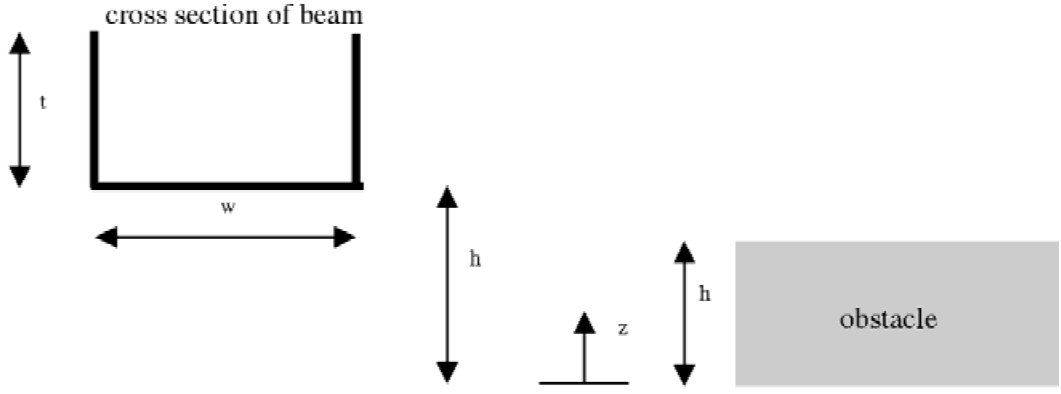


Fig. 7. A cross section of a hood panel beam.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

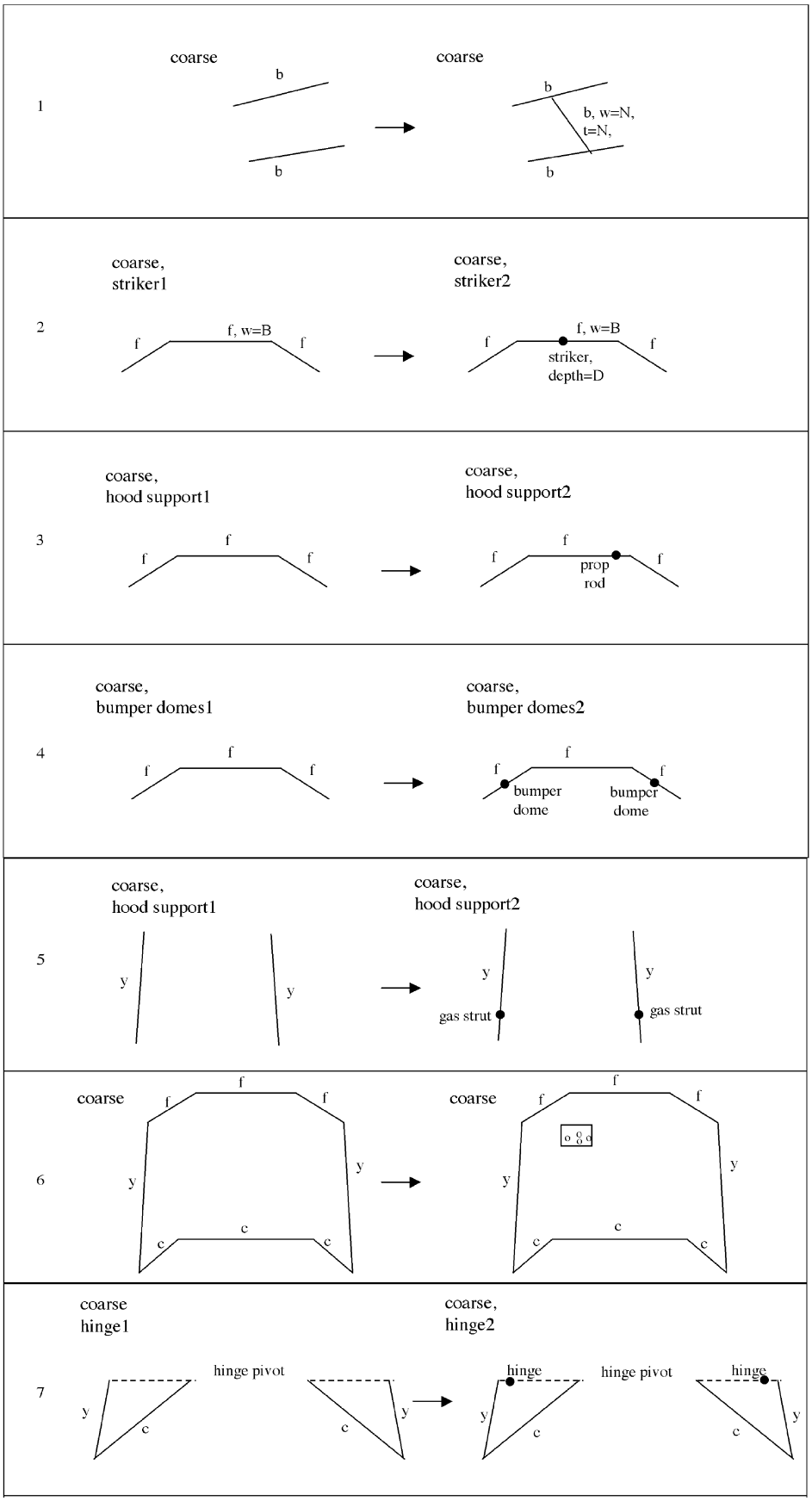


Fig. 8. Rules 1–7 apply to the creation and placement of hood components and obstacles in the coarse stage of design.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

the striker along with a label *depth*, whose numeric value stores the location of the striker on the width of the front beam and changes the global label *striker1* to *striker2* to prevent reapplication of the rule. Rule 3 adds a point to designate the location of the prop rod and changes the *hood support1* global label to prevent rule reapplication. Rule 4 locates the bumper domes and changes the *bumper domes1* global labels to prevent rule reapplication. Rule 5 adds gas struts to the side beams of the inner hood and also changes the *hood support1* label. Rule 6 allows the user to add obstacles to the design. This allows the designer to model additions to the engine package within the confines of the shape grammar. Rule 7 places the hinges along the hinge pivot line.

Rules 8–10 (Fig. 9) are powerful rules, which take advantage of shape emergence. They can be used to rectify situations where a beam is interfering with an obstacle. From the interference, a pair of intersecting lines emerges, one line labeled *b* and the other line labeled *o*. The left-hand side of each of these rules features the intersection of a line labeled *b* and a line labeled *o*, which, when using a parametric shape recognition scheme, allows these rules to be matched to any parametric equivalent of the intersecting lines. Section 5 contains further discussion on the shape recognition strategy, which allows for these rules to cover instances of interference comprehensively. Rule 8 can be applied when a beam intersects an obstacle and the value of the height label on the beam is less than or equal to the value of the height label on the obstacle. Application of rule 8 moves the endpoint of the intersecting beam along the connecting line segment in order to rectify the interference. Similarly, rule 9 can be applied in the same situation as rule 8; however, application of rule 9 removes the interfering beam and replaces it with two beams to avoid the obstacle. The final method for correcting interference in the coarse design stage is to increase the height of the beam in order to have it pass over the obstacle. The designer would undoubtedly have restrictions on how high the beams could be moved. This situation is addressed in rule 10.

During the design process some features may need to be adjusted. Several rules were written to allow this change to be modeled within the confines of the grammar. Rules 11–14 (Fig. 10) allow features to be moved to new positions. The four rules work very similarly, recognizing features that are represented with labeled points and moving them to new locations on the same line on which they were originally located. Rule 15 moves the hinge pivot line, as well as the hinge pivot points, vertically to a new location.

Rules 16–18 (Fig. 11) change parameter values of beams by changing label values. Rule 16 allows the designer to change the beam width, rule 17 allows changes in the beam height, and rule 18 allows changes in the thickness. Application of rule 19 (Fig. 11) allows the removal of a beam.

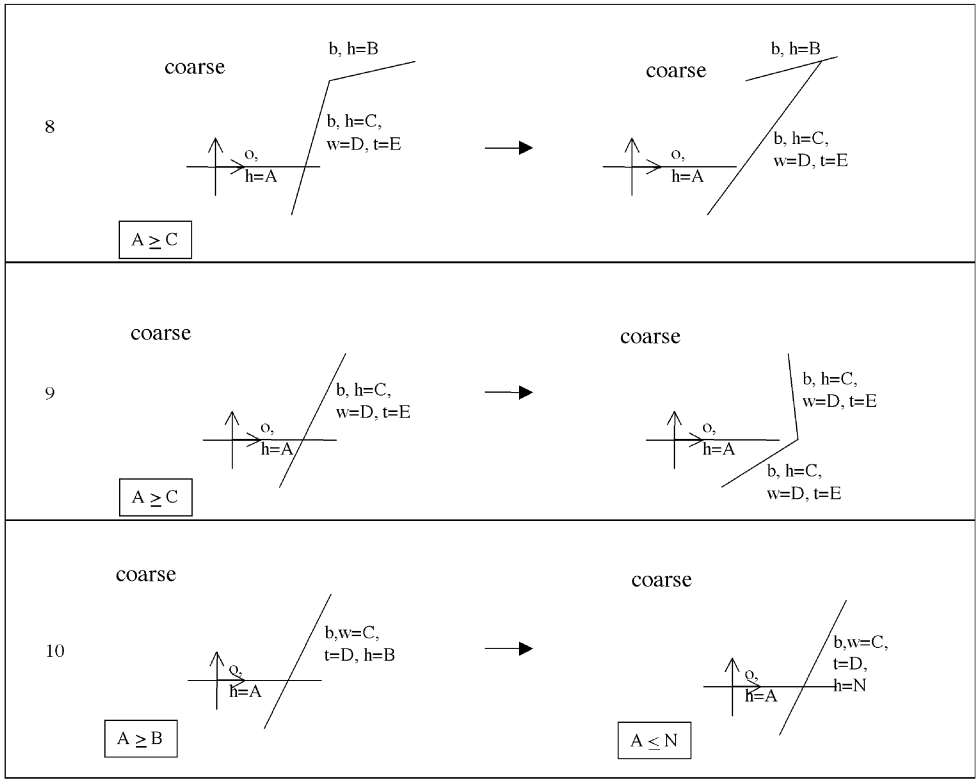


Fig. 9. Rules 8–10 are for correcting obstacle interference.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

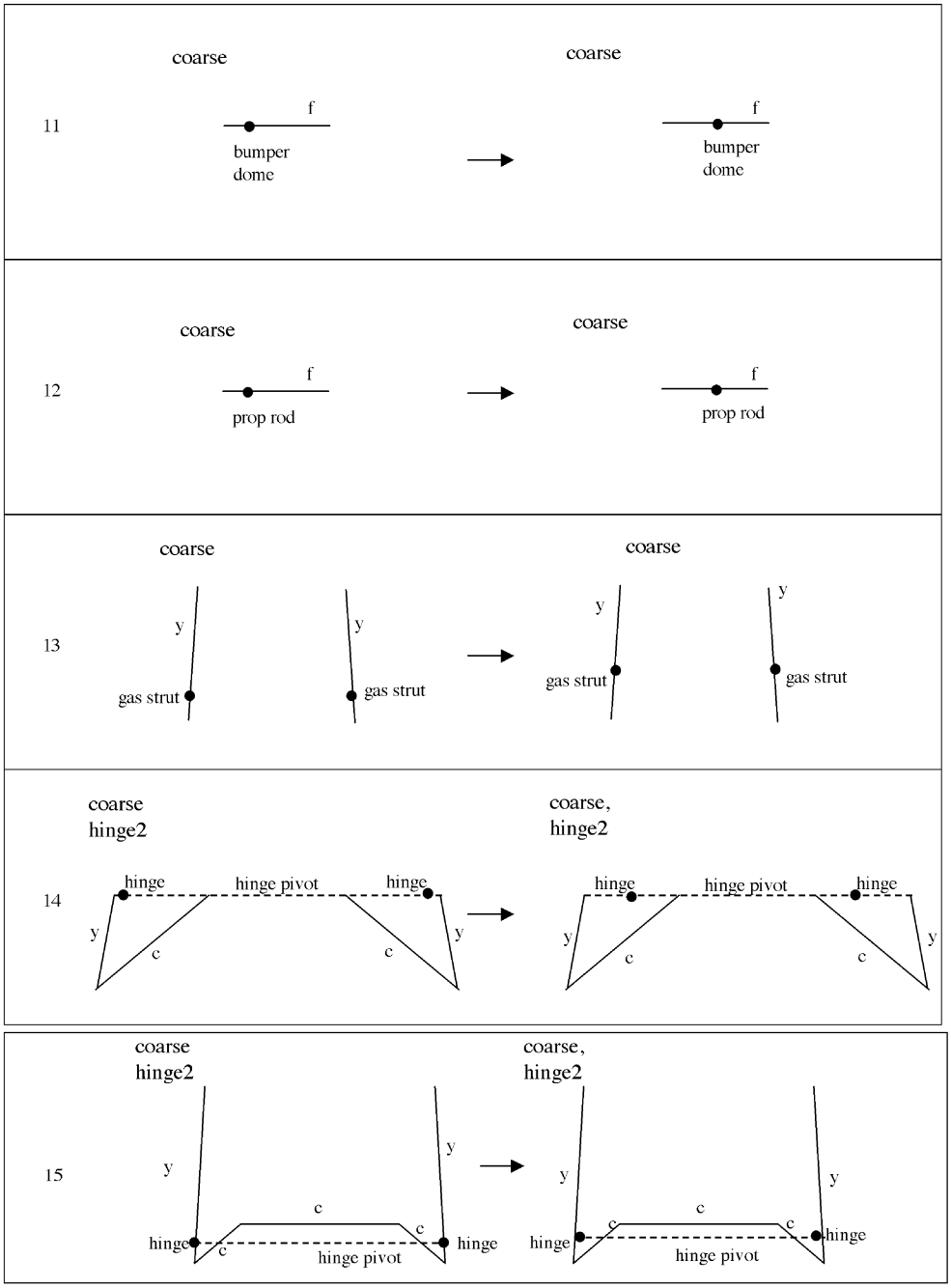


Fig. 10. Rules 11–15 are the coarse stage feature adjustment rules.

Rules 20 and 21 (Fig. 12) are used to adjust obstacle parameters and locations. Rule 20 allows existing obstacles to be moved to new locations, and rule 21 changes the height of an object.

Rules 22–24 (Fig. 13) allow the designer to transition from the coarse rules to the fine rules (Rule 22) and from fine back to coarse (Rule 24). Rule 23 removes the geometric representation of the beam width.

Rules 25–27 (Fig. 14) let the designer work in the fine design regime. If there is fine obstacle interference, application of rule 25 places a notch in the beam so that the center line does not have to be moved. Rules 26 and 27 allow the designer to geometrically represent the width of a beam. In the coarse design stage, only a label represents the beam width, but the geometric representation of the beam width in the fine stage lets the designer perform a finer check for interference. Finally, if the appropriate global labels are present, rule 28 (Fig. 14) can be applied and the design is completed.

The 28 rules of this shape grammar can be applied in any order, providing that a match for the left-hand side of the rule can be found.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

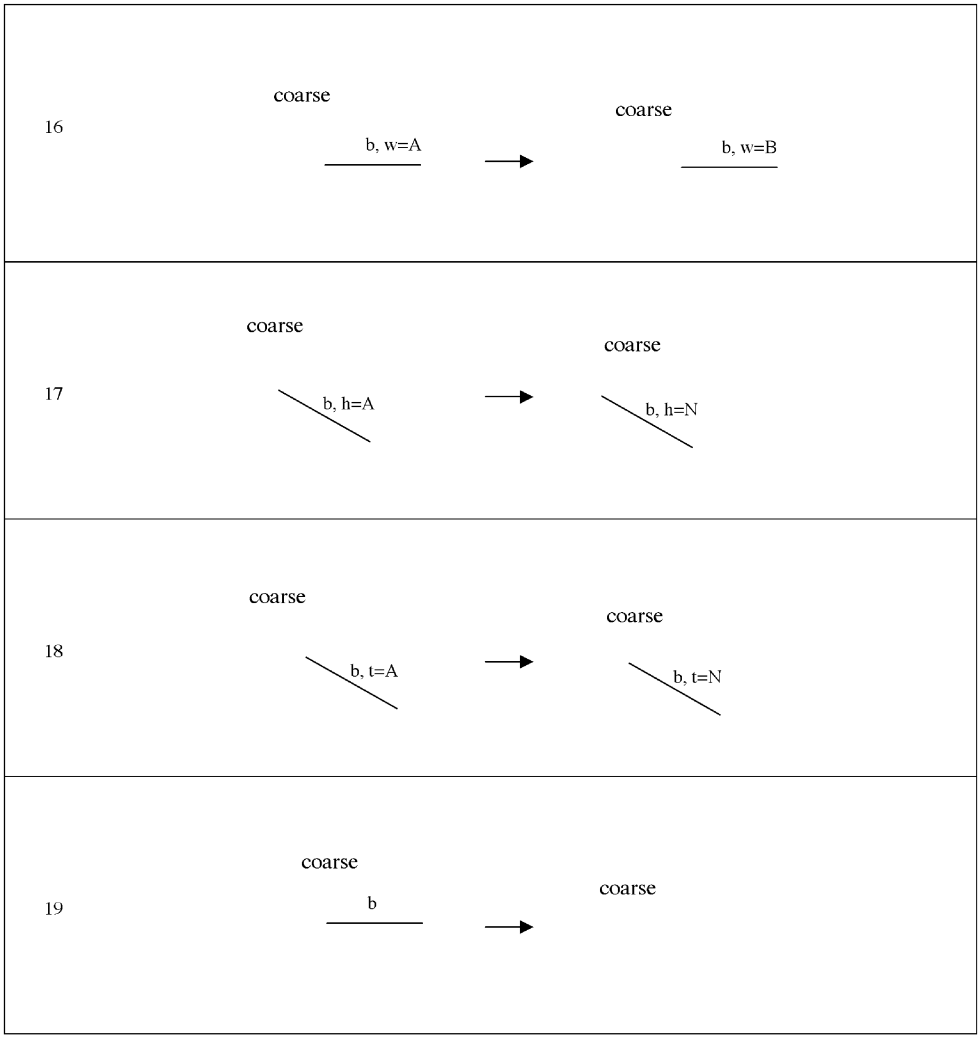


Fig. 11. Rules 16–19 apply to beam parameter adjustment rules and beam removal.

4. SHAPE GRAMMAR INTERPRETER

Shape grammars serve as useful guides to a designer when used on paper but can be even more valuable if they are implemented and used as a computer tool. As shown in the framework in Figure 1, the interpreter takes the rules of the shape grammar and performs three key tasks for implementation. First, it determines if a rule can be applied by searching for an instance of the rule's left-hand side shape in the working shape. Second, the interpreter determines where that rule can be applied by finding all instances of the rule's left-hand side shape in the working shape. Third, it possesses the functionality to apply a rule by working interactively with a user or optimization routine. Traditional interpreters (Krishnamurti, 1980) have employed a shape matching scheme wherein the left-hand side shape has been transformed using a combination of translation, rotation, and scaling to find a match in the working shape. Unfortunately, the shape matching schemes of traditional interpreters do not often meet the needs of engineering grammars, which require parametric handling of shape. In some complex grammars, each rule is accompanied by text explaining what the writer intends as a shape match. This is necessary because the rule writer's intentions do not always match the interpreter's shape matching scheme. This ultimately leads to three unfortunate consequences: rules do not rely on shape recognition for determining rule applicability and instead rely on label matching, thereby eliminating opportunities for shape emergence, the creative potential harnessed within shape manipulation; rules have to be implemented individually, requiring an unacceptable amount of time, possibly 6–12 months for a large rule set (typically around 50 rules); and rule sets are generally larger, requiring multiple rules to model subcases covered by a single parametric rule that uses shape recognition.

As an example of the limitations of shape matching using translation, rotation, and scaling of entire shapes, consider rule 9 (Fig. 15). The rule is to be applied to any situation

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

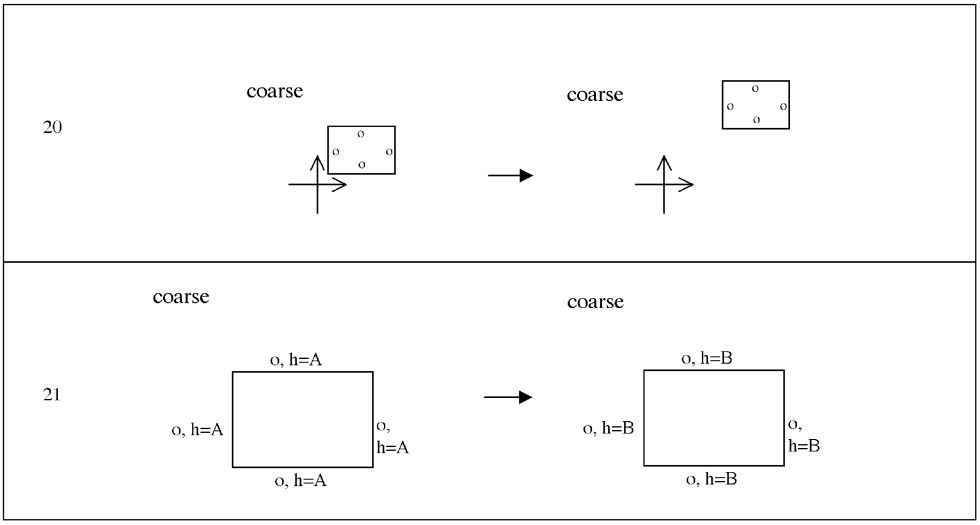


Fig. 12. Rules 20–21 are for adjusting obstacle parameters and locations.

where a line labeled *o* and a line labeled *b* intersect. An attempt to match the shape using only translation, rotation, and scaling will exclude desired matches such as properly labeled lines that meet at a different angle or lines with a different ratio of lengths (Fig. 16). Shape grammar interpreters employing this shape matching scheme would be insufficient.

However, the hood grammar was implemented with a shape grammar interpreter created by us (McCormack & Cagan, 2000, 2001) that gives the designer freedom to use shape recognition where matching does not have to be uniformly applied to an entire shape and is customizable to the user's needs. This interpreter allows not only the hood grammar but also other grammars to be implemented very quickly

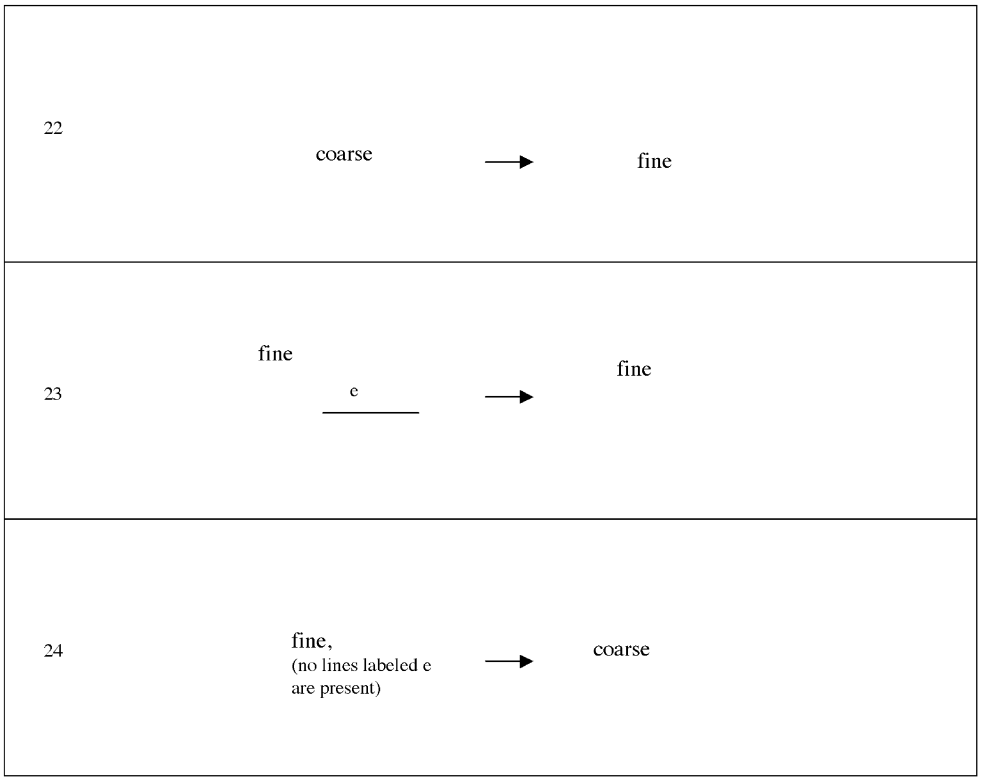


Fig. 13. Rules 22–24 apply to transitions between the fine and coarse stages of design.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

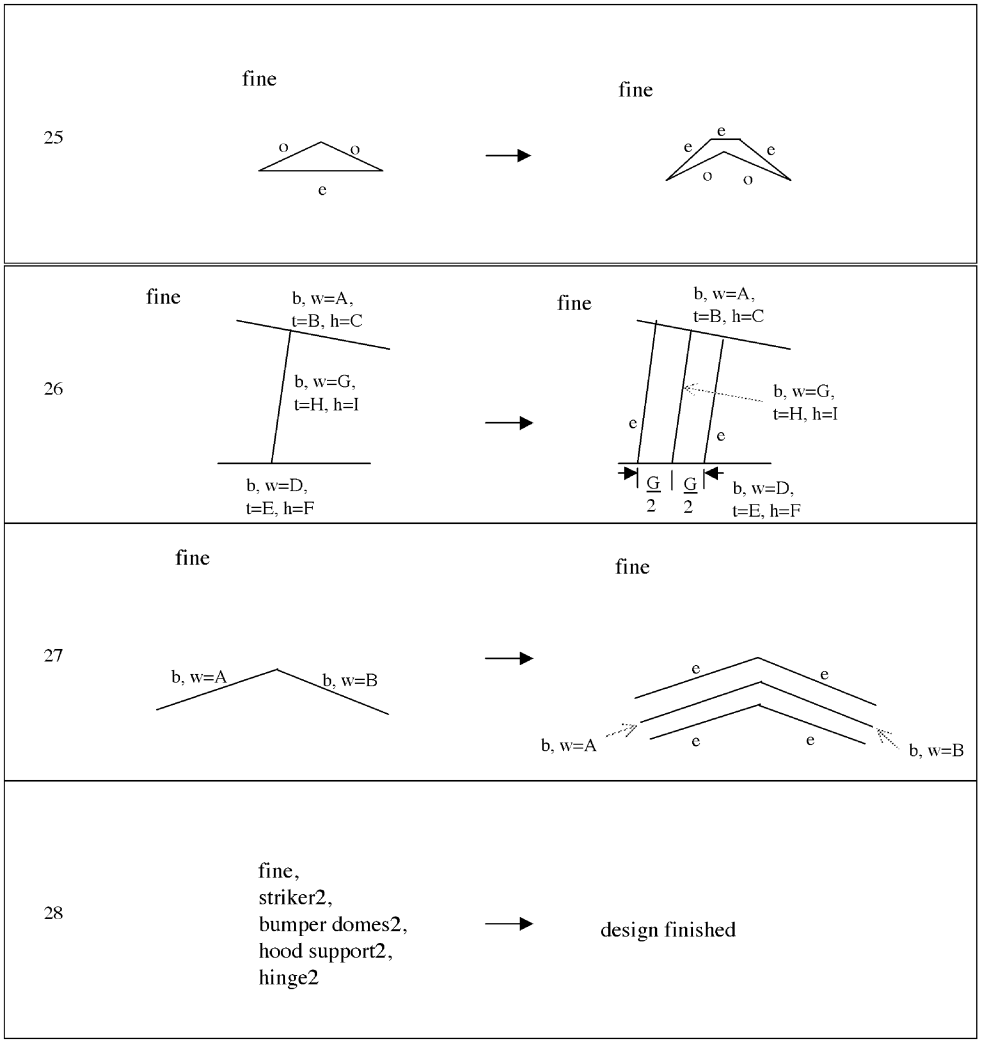


Fig. 14. Rules 25–28 are the rules for the fine stage of design and termination.

(hours vs. months), making the use of a shape grammar as a computational tool for design more practical.

Our shape recognition process begins by dividing the left-hand side shape into a set of subshapes for which the sum of the subshapes equals the whole original shape. The division into subshapes through the user-defined hierarchy allows portions of the original shape to be transformed separately and differently in an effort to find equivalent shapes that match according to the designer's concept of a shape match. A section of the hood in Figure 17 is broken down into its subshapes as an example of the hierarchy.

The shape extracted from the inner hood is decomposed into three subshapes. These subshapes match the first three levels (s_1, s_2, and s_3) of our default hierarchy presented

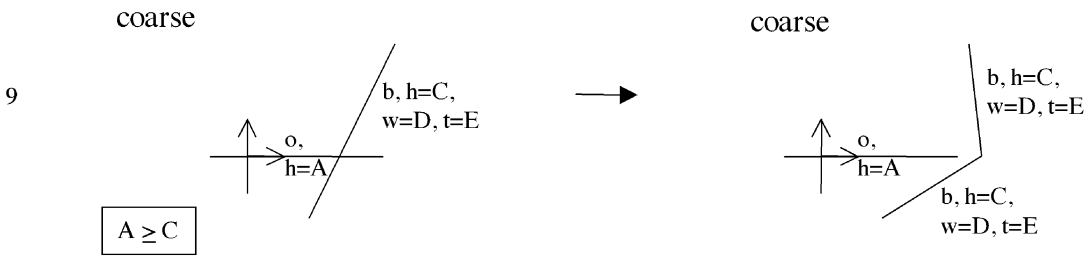


Fig. 15. Rule 9 is from the inner hood shape grammar.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

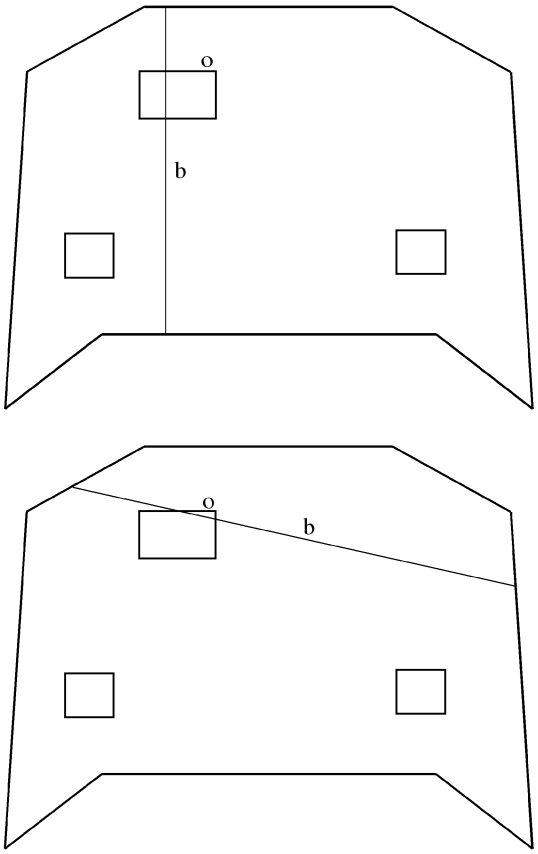


Fig. 16. Instances that cannot be matched with traditional interpreters.

previously (McCormack & Cagan, 2000, 2001). Shape s_1 is formed from line segments that intersect perpendicularly and are the same length or lines that are symmetric to more than one line that are not parallel. The sides of the square are members of the s_1 grouping in Figure 18. Shape s_2 is composed of line segments that intersect perpendicularly, are symmetric to one line, or are symmetric to more than one line that are parallel. In the example, the subshape s_2 is composed of the sides of the isosceles triangles. The s_3 shape contains lines that are to be matched with an equal number of line segments with the same connectivity. The s_3 subshape in Figure 18 is formed from the remaining line segments. The fourth level of the default hierarchy, the s_4 shape, contains line segments that have no apparent geometric relations to other line segments. There are no s_4 lines in the example.

By placing sets of line relations (symmetry, right angle intersections, etc.) together into groupings based on levels of constraint, sets of transformations are determined that form the default hierarchy. The transformations for each level of the hierarchy are chosen to transform the shapes at that level without destroying the key line relations. The line relations and their importance are chosen based on its engineering and design significance, as determined by the authors and verified as relevant through available published shape grammars. The default hierarchy is composed of four groupings that are associated with the four subshapes from the most constrained to the least (s_1, s_2, s_3, and s_4). For example, with the sides of a square (s_1), any anisotropic scaling or shearing would destroy the symmetry or the perpendicular intersection of the line segments. The transformations associated with the other groupings are chosen for similar reasons. It is important to note that other hierarchies may be created to meet the specific needs of the shape grammar writer. Additionally, the presentation of the hierarchical decomposition in this paper is done in a manner that implies that the shapes that define rules are decomposed in an effort to determine the intentions of the rule writer. In fact, the rule writer has knowledge of and even

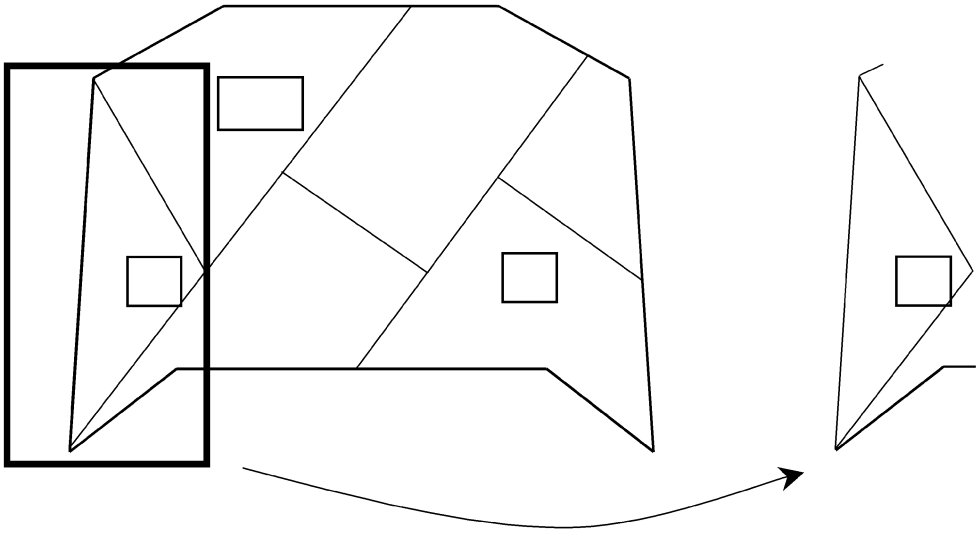


Fig. 17. A section of an inner hood is extracted for decomposition.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

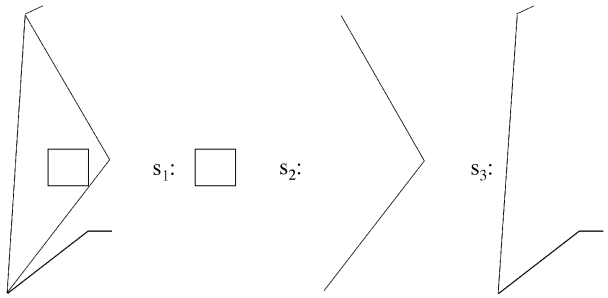


Fig. 18. The example shape and each of its subshapes.

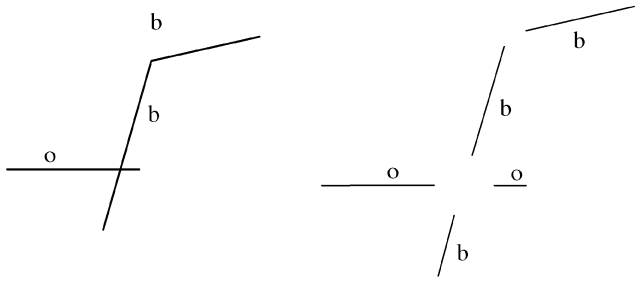


Fig. 20. The left-hand side shape of rule 2 broken down into line segments.

determines the hierarchy while writing the grammar rules, allowing the writer to specify an entire shape and its decomposition based on that hierarchy of spatial relations.

The shape search begins with the most highly constrained subshape and progresses through each of the remaining subshapes until a search has been conducted for each. After the first subshape search, matching instances are subtracted from copies of the original shape. The next subshape search is performed in the altered copy, so as to not match line segments in multiple searches. The points that are shared between the two subshapes are marked with a label so that an entire shape can be recreated after the shape matches have been found.

Having established this default hierarchy, it is important to reiterate the point that the hierarchy is customizable to the designer's needs. The type of transformations to be used for each subshape grouping is up to the designer as long as the set of shape decompositions is complete. Our earlier work (McCormack & Cagan, 2000, 2001) provides more detail on the interpreter.

The application of rule 8 from the inner hood grammar demonstrates the shape matching process for a shape that contains line segments in the s_3 grouping. The bold lines in Figure 19 show where a subshape matching the left-hand side of rule 8 is located. The subshape is matched by finding line segments with the same labels and an equivalent connectivity. Figure 20 shows the left-hand side of rule 8 and the same shape broken down into the set of smallest line segments based on actual line intersections. By extracting the matching subshape in the sample hood and also breaking down this subshape, the equivalent connectivity between line segments is apparent. The search to find the subshape is performed by a complete search through the shape, which can be expedited by searching for line segments with more rare labels and for line segments with multiple shared points first, thereby identifying the location of potential subshapes.

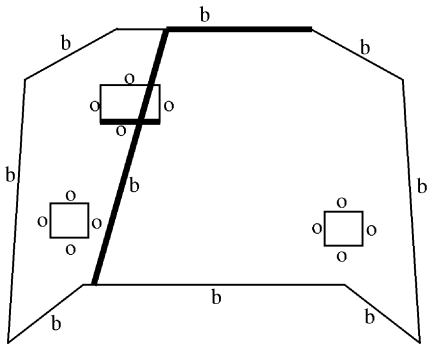


Fig. 19. A beam interfering with an obstacle is corrected through application of a rule.

Application of rule 8 to the bold subshape (Fig. 21) produces the corrected hood in Figure 22. Despite the fact that the designer did not add the bolded obstacle line and the bolded beam line individually, the shape that matches the left-hand side of rule 8 emerged. Without a parametric shape recognition scheme, however, no single general rule could be written to account for the variety of possible obstacle intersections that could occur or to recognize the emerging shape.

5. SAMPLE HOOD GENERATION

The inner hood shape grammar rules can now be used to create valid hoods by working interactively with the interpreter. The interpreter provides the user with the rules that can be applied, then the user chooses one and is presented with all the locations where the chosen rule can be applied. The rule is then applied, with the user providing any parameters that may be necessary.

Several steps in an example design sequence are shown in Figure 23. The process begins with the initial shape in Figure 23(a). The application of rules 2, 3, 4, and 7 adds labeled points representing the striker, prop rod, bumper domes, and hinges in Figure 23(b). Figure 23(c) is the design after four applications of rule 1, which add four interior beams. Rule 1 is written such that the interpreter will match any two properly labeled line segments because of the lack of geometric information in the rules left-hand side. Next the grammar is used to model a changing engine package. Rule 20 in Figure 23(d) is used to move the obstacle to its new location, creating interference. The interference is identified through shape recognition, and rule 8 [Fig. 23(e)] is then used to alter the beam structure and avoid the moved

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

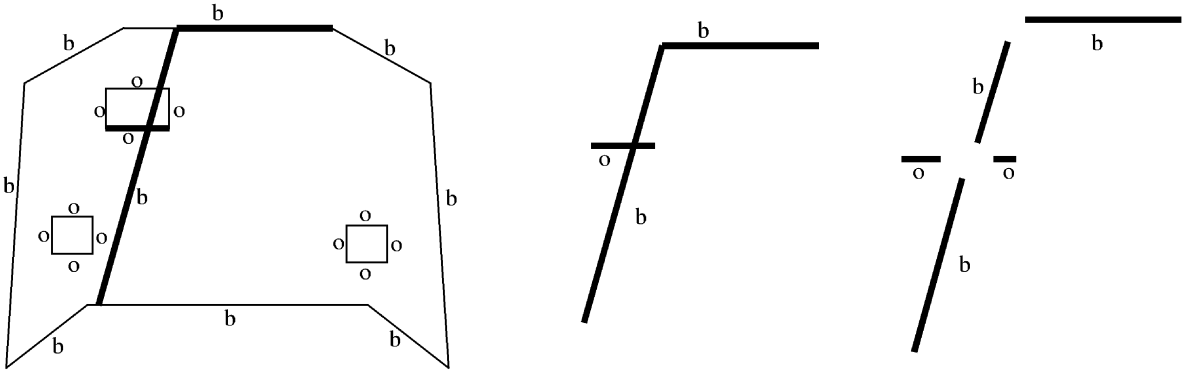


Fig. 21. A matching subshape found in the sample hood broken down into line segments.

obstacle. Again, note the power of shape recognition and the benefits of our interpreter in directly acting on the emergent interference.

The design then moves to the fine design stage in Figure 23(f) using rule 22. Multiple applications of rules 26 and 27 create a geometric representation of beam width, and a check is then performed for interference among beams and obstacles. Most labels are omitted in Figure 23(f), to make the diagram clearer. Repeated application of rule 23 removes the geometric representation of beam width in Figure 23(g). Finally the design is completed by applying rule 28, which adds the *design completed* label [Fig. 23(h)]. By using the rules, the designer is assured of creating a hood design possessing the necessary functional components. However, the knowledge behind the arrangement of the internal beam structure is not present. Instead, only the means to add beams and correct interference problems among beams and obstacles exists in the grammar, leaving the designer (human or computer) to interact with the grammar to make decisions that determine the feasibility and quality of the beam structure.

6. OPTIMIZATION

As opposed to using the grammar interactively through the interpreter, the grammar can be used by an optimization scheme that applies rules and instantiates parameters of the shape grammar using the interpreter. Cagan and Mitchell (1993) introduced the idea of optimization coupled with shape grammars in which simulated annealing was applied to a shape grammar to create optimally directed shapes. The simulated annealing, or "shape annealing," technique was also applied to shape grammars whose rules described the construction of truss structures (Shea & Cagan, 1997) and transmission towers (Shea & Smith, 1999). The simple nature of these grammars required a directed search to move through the design space effectively, eliminating infeasible alternatives while moving toward an optimal design.

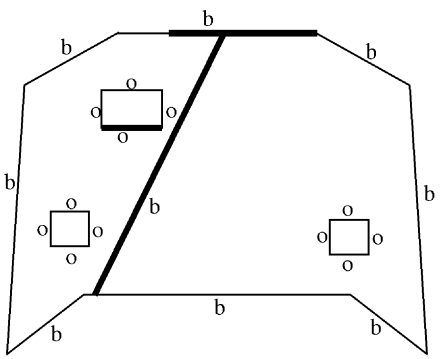


Fig. 22. The result of the application of rule 8 to the shape in Figure 19.

More knowledge intensive grammars, through their own definition, eliminate some, if not all, infeasible designs and should rely on a more generative approach for optimization (Cagan, 2001). We have begun to explore the feasibility of an agent-based approach that can take advantage of the level of knowledge present in the grammar to more selectively create designs. For our work we define agents as simple strategies that have the ability to perceive and affect their environment (Campbell et al., 1999). Through the simple actions of many agents a complex process can take place such as the creation of a product. Each agent is coupled to a shape grammar rule, which provides the knowledge that the agent uses when affecting the design through the interpreter.

The amount of computational time potentially required for performing detailed shape matching is significant enough that it would be advantageous to avoid using an approach that could potentially require tens of thousands of iterations such as a robust simulated annealing method using a dynamic annealing schedule. We introduce a variation of the A-Design agent-based approach (Campbell et al., 1999) that uses a short geometric schedule to control the level of stochastic search. The stochastic element of the scheme permits infeasible designs to be created en route to an improved final design. This method balances search with the use of knowledge inherent in the hood grammar.

The method works as follows. Two types of agents were used: manager and instantiation agents. Manager agents select which rule to apply by first examining the design, com-

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

https://doi.org/10.1017/S089000604021640 2X Published online by Cambridge University Press

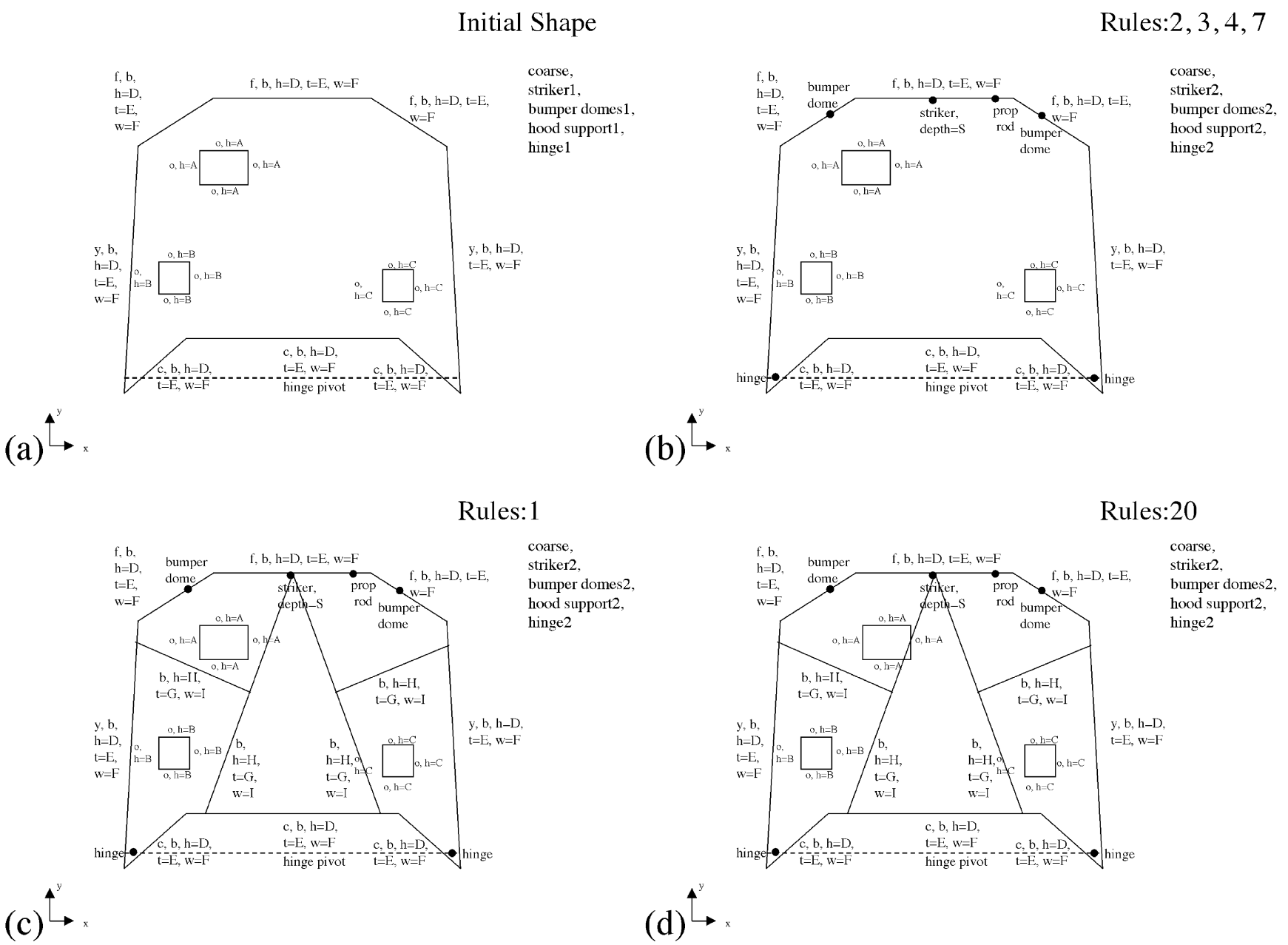


Fig. 23. Sample hood generation.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

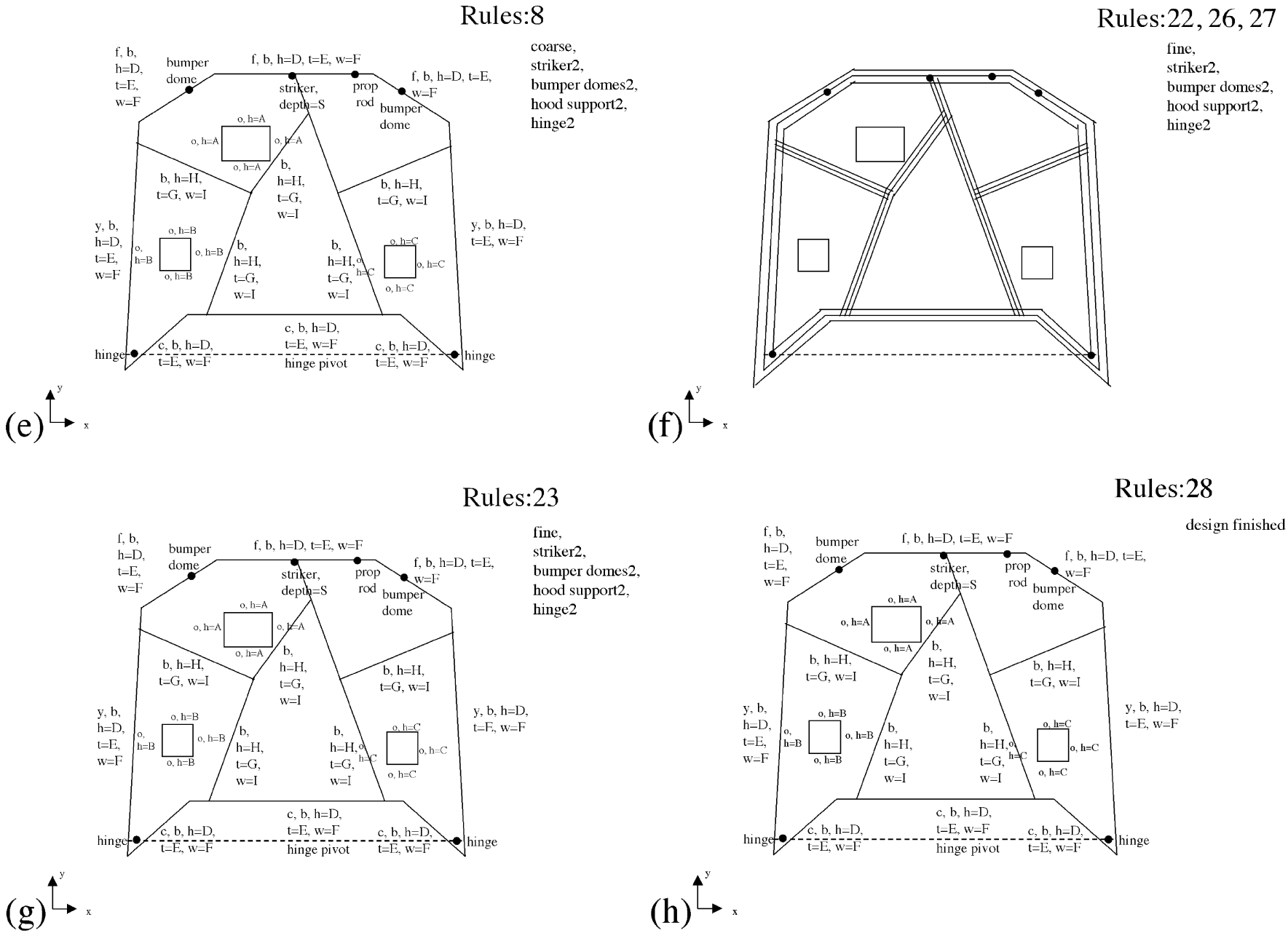


Fig. 23. Continued.

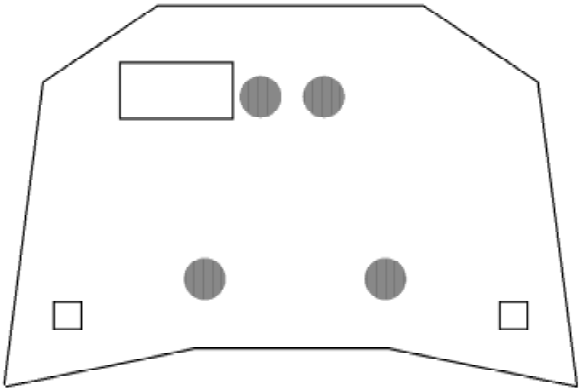


Fig. 24. The four key structural zones.

puting the values of the objective and penalty functions, and then, based on that data, deciding what facet of the design needs to be addressed. The manager agent uses a bookkeeping system to track how each rule affects the objective and penalty functions and uses that history to probabilistically determine which rule is more likely to address the current need. Each shape grammar rule is associated with an instantiation agent that, when called upon, can work with the shape grammar interpreter to apply its rule, providing the interpreter with the parameters that it needs to complete the rule application. The instantiation agent sets parameters by having knowledge of the objective function and tests its own effect on the objective function at coarse increments. After a rule is applied, the manager agent keeps the new design if it is better than the previous and probabilistically selects to keep a poorer new design according to the annealing schedule.

The evaluation of the hood consists of determining the weight of the hood and the quality of the placement of the structural beams. The lengths of the beams are summed and serve as the weight. The quality of beam placement is determined by the location of beams with respect to four key zones (Fig. 24). If a beam does not pass through each of the key zones, a penalty is assessed. A penalty is also levied for each beam that interferes with a portion of the engine, battery, or other under-hood obstacle. Some preliminary designs that were created from this exploratory optimization work are shown in Figure 25. These hoods were generated within 100 iterations (or attempted rule applications) performed by the agents. It is worth noting that the agents created the hood with a symmetrical beam configuration [Fig. 25(b)] in only 4 iterations. The symmetrical design in Figure 25(b) avoids all penalties by avoiding obstacle interference and placing beams through all four key zones. The hood in Figure 25(a) is a lesser quality design; it avoids interference but only places beams in three of the key structural zones. The other designs in Figure 26 were produced from other runs.

The objective function values of the designs in Figure 25(a,b) are 904 and 641, which represent a typical design and the best design, respectively. The mean value of the objective function of the best design per run is 938, where the best value was achieved on average by the 49th iteration. Objective function values often increased in the second half of the run due to the simplicity of the good solutions as well as the stochastic nature of the approach. The optimization work performed in this project could be improved and built upon by addressing the simulated annealing or agent portion of the approach, however, as a first effort, it met our needs in demonstrating the automation loop of the grammar-based design tool framework (Fig. 2).

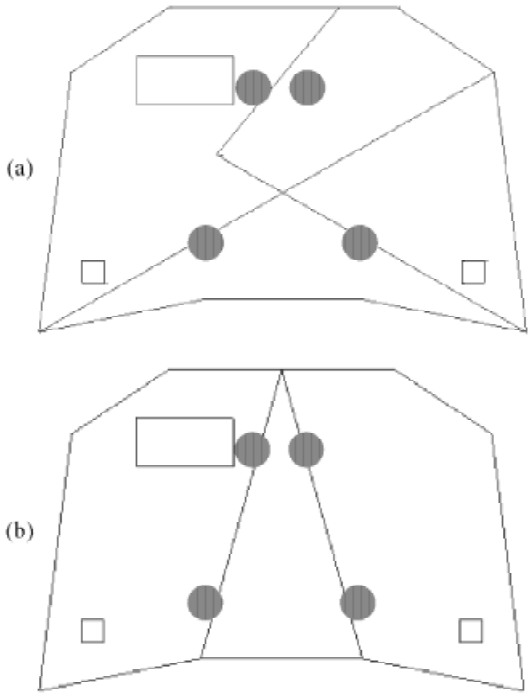


Fig. 25. Preliminary designs of hoods created through automated generation.

7. CONCLUDING REMARKS

This work demonstrates the potential effectiveness of the shape grammar based computational tool. Our work and previous efforts show that a shape grammar is an effective way to represent the language of a class of parts and products. The interpreter makes it possible to implement shape grammars with a customizable, parametric shape recognition algorithm that allows a user or an optimization scheme to work interactively with the grammar. By allowing the designers to work with parametric shape grammars whose rules employ shape recognition, the ability to create novel designs through shape emergence is possible.

The level of detail represented in the grammar is a function of what the designers wish to accomplish with the grammar and current limitations in the types of shapes supported by the shape grammar interpreter. If the designers wanted

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

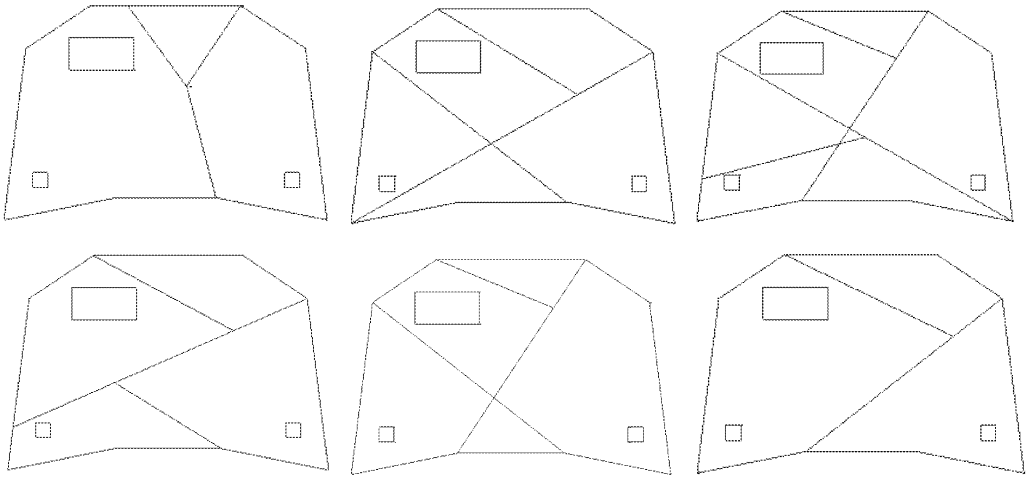


Fig. 26. Additional runs of hoods created through automated generation.

to include additional details, this would be possible, but for our purposes, the level of detail was appropriate because we showed that the agents coupled with the shape grammar interpreter could create hood designs while dealing with hood packaging issues and providing sufficient structural support within the context of the shape grammar.

The automation work was exploratory but served as a test bed for future optimization problems based on shape grammars using a similar approach. The current objective function is useful for the early stages of inner hood panel design. Downstream analysis or refined optimization would require a more detailed structural evaluation.

It is also apparent that an in-depth optimization must include an objective function that evaluates the true quality of the design. The shape grammar rule writer should keep this in mind and should include sufficient detail in the grammar used in downstream design so that the quality of the designs produced can be properly evaluated. In order to include sufficient detail the grammar writer may have to move to 3 dimensions, include curves, or simply include additional 2-dimensional detail. The concepts in this paper concerning shape grammars, implementation, and potential optimization as a computational tool can be applied to the design of other car panels, car parts, or other products.

ACKNOWLEDGMENTS

The authors would like to thank the National Science Foundation under Grant DMI-9713782 and General Motors for their support of this work. The authors would also like to thank Jim Elshoff and Mary Pickett for their input and discussions on the hood grammar.

REFERENCES

Agarwal, M., & Cagan, J. (1998). A blend of different tastes: The language of coffee makers. *Environment and Planning B: Planning and Design 25*, 205–226.

Agarwal, M., & Cagan, J. (2000). On the use of shape grammars as expert systems for geometry based engineering design. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing 14*, 431–439.

Agarwal, M., Cagan J., & Stiny, G. (2000). A micro language: Generating MEMS resonators using a coupled form-function shape grammar. *Environment and Planning B: Planning and Design 27*, 615–626.

Brown, K.N., McMahon, C.A., & Sims Williams, J.H. (1994). A formal language for the design of manufacturable objects. In *Formal Design Methods for CAD (B-18)* (Gero, J.S., & Tyugu, E., Eds.), pp. 135–155. Amsterdam: North–Holland.

Cagan, J. (2001). Engineering shape grammars: Where have we been and where are we going? In *Formal Engineering Design Synthesis* (Antonsson, E.K., & Cagan, J., Eds.), pp. 65–92. Cambridge, UK: Cambridge University Press.

Cagan, J., & Mitchell, J.W. (1993). Optimally directed shape generation by shape annealing. *Environment and Planning B 20*, 5–12.

Campbell, M., Cagan, J., & Kotovsky, K. (1999). A-design: An agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design 11*, 172–192.

Krishnamurti, R. (1980). The arithmetic of shapes. *Environment and Planning B: Planning and Design 7*, 463–484.

McCormack, J., & Cagan, J. (2000). Supporting designer's hierarchies through parametric shape recognition. *Proc. 2000 ASME Design Engineering Technical Conf. and Computers in Engineering Conf.: Design Theory and Methodology* (DETC00/DTM-14555), Baltimore, MD, September 10–13.

McCormack, J., & Cagan, J. (2001). Supporting designer's hierarchies through parametric shape recognition. *Environment and Planning B.*

Pugliese, M., & Cagan, J. (2001). Capturing a rebel: Modeling the Harley–Davidson brand through a motorcycle shape grammar. *Research in Engineering Design 13*, 139–156.

Shea, K., & Cagan, J. (1997). Innovative dome design: Applying geodesic patterns with shape annealing. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing 11*, 379–394.

Shea, K., & Smith, I. (1999). Applying shape annealing to full-scale transmission tower re-design. *Proc. 1999 ASME Design Engineering Technical Conf. and Computers in Engineering Conf.: Design Automation Conf.* (DETC99/DAC-8681), Las Vegas, NV, September 12–15.

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

Stiny, G., & Gips, J. (1972). Shape grammars and the generative specification of painting and sculpture. *Information Processing 71*.

Jonathan Cagan is a Professor of mechanical engineering at Carnegie Mellon University with appointments in the School of Design, Biomedical Engineering Department, and

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press

Computer Science Department. His research, teaching, and consulting are in the areas of design theory, methodology, automation, and practice. He received his BS in 1983 and MS in 1985 from the University of Rochester and his PhD in 1990 from the University of California at Berkeley, all in mechanical engineering. Dr. Cagan is the recipient of the National Science Foundation's NYI Award and the Society of Automotive Engineer's Ralph R. Teetor Award for Education. He is a member of the Phi Beta Kappa, Tau Beta Pi, and Sigma Xi National Honor Societies; a Fellow of ASME; and a member of IDSA, AAAI, SAE, and ASEE. Dr. Cagan is a registered professional engineer.

Jay McCormack is a PhD candidate in the Department of Mechanical Engineering at Carnegie Mellon University and is advised by Jonathan Cagan. He received a BS in 1998 and an MS in 2000, both from Carnegie Mellon. Mr. McCormack is a member of the Honor Society of Phi Kappa Phi.

https://doi.org/10.1017/S089006040216402X Published online by Cambridge University Press