UNIVERSITY OF CAMBRIDGE

PERFORMANCE-BASED COMPUTATIONAL SYNTHESIS OF PARAMETRIC MECHANICAL SYSTEMS

Alex C Starling
St John's College

January 2004

A dissertation submitted for the Degree of Doctor of Philosophy
Cambridge University Engineering Department

In memory of

MATTHIAS WAGNER

* 7 JUNE 1977 † 3 DECEMBER 2003

Taken too soon

Declaration

Except where otherwise stated, this thesis is the result of my own research and does not include the outcome of work done in collaboration.

This thesis has not been submitted in whole or in part for consideration for any other degree of qualification at this University or any other institute of learning.

This thesis contains 94 figures, 27 tables and less than 52,000 words.

Alex C Starling
St John's College
Cambridge
January 2004

Abstract

Keywords: computational synthesis, design search, design synthesis, grammatical design, parallel grammar, performance-based design, vehicle transmission design

This research seeks to develop a synthesis formalism to aid and enhance the design of mechanical systems by harnessing the power of the computer. For computer-assisted generative design to be effective for exploring purposeful design possibilities to a given specification, many tasks that to date are performed manually or as part of designer-intensive computer-based processes must be approached in new ways.

This research contributes a new type of production system, a parallel grammar for mechanical systems, developed using a Function-Behaviour-Structure representation, to generate and modify a variety of designs. Geometric and topological constraints are used to bound the design space, termed the language of the grammar, to ensure the validity of designs that can be generated with the grammar. Four case studies are considered, namely the design domain of mechanical clocks and watches, the redesign of an electromechanical camera winding mechanism, power drill design and the generation of alternative vehicle gearbox configurations. For verification purposes, the parallel grammar is used by hand to recreate existing designs. Computational generation of novel design configurations is driven by performance-based evaluation of designs using geometry-based metrics and behavioural analysis of automatically generated simulation models. Multi-objective stochastic search, in the form of a hybrid pattern search developed as part of this research, is used to generate Pareto sets of optimally directed designs. The work is validated through an investigation into the generation of novel transaxle gearbox designs in collaboration with an automotive power transmission design company.

Acknowledgements

The following people and organisations have provided and/or facilitated funding for various parts of this research. I am extremely grateful for this support.

Dr Kristi Shea
Dr R E McConnel
Mr Malcolm Shirley
The American Association for Artificial Intelligence
The National Science Foundation
Department of Engineering, Cambridge University
The Leverhulme Trust
The Royal Academy of Engineering
Dr P John Clarkson
St John's College, Cambridge
The Engineering and Physical Sciences Research Council
The Newton Trust
The Royal Commission for the Exhibition of 1851

Many other people have assisted with contributions of an academic or technical nature, notably Dr Rob Bracewell, Dr Chris Vale and Francesca Bolognini. I am indebted to Dr Peter Poon and Barry James at Romax Technology for sharing their vision of the future; thanks also to Kevin Hewitt at Black and Decker. Andrew Flintham provided and maintained the infrastructure that made possible much of the implementation side of this research. Above all, my thanks go to Dr Kristi Shea for her acumen and drive, and for setting out high standards of achievement to strive for.

Finally, this thesis would not have happened without the support of my friends and family.

Thank you all.

Contents

Figures

Tables

1. Introduction

In an age where computers pervade much of our lives, it is perhaps surprising that this 'rise of the machines' has been a very recent trend. Almost a century passed after the death of Charles Babbage, the 'father of computing', in 1871 before truly automated computing began to take a hold of Western civilisation. Since then, silicon-based central processing units (CPUs) have become extremely fast calculating machines that people rely on, directly and indirectly, as they go about their daily lives. For engineers, the computer is a key tool, used for enhanced sketching, i.e. Computer Aided Design (CAD), analysis and optimisation, as well as a myriad of other purposes[1]. The success of these approaches lies in harnessing the power of the computer to rapidly perform routine tasks that would otherwise take too much time to carry out by hand.

While computational methods and tools have greatly influenced our ability to model and analyse potential designs, they have not yet contributed greatly in terms of design *synthesis*, i.e. the generation of novel product and artefact designs. Despite the proliferation of computer tools for all aspects of engineering design, it seems that the task of synthesising new ideas and creating innovative and original concepts has remained almost the exclusive burden of the human mind. Mapping design specifications to tangible embodiments, i.e. the synthesis of new design ideas, remains, in this age of the computer, a manual task. The core question underlying this research is whether computational methods can be developed that can stand alongside the mechanical engineer as a 'synthesis partner' to further the development of innovative design generation.

[1] i.e. for machine tools, robotics, knowledge management, networking, communication, …

The aim of this research is summarised in the thesis statement in Figure 1-1.

> *'The aim of this research is to enhance the design of parametric mechanical systems through the development and use of computational synthesis methods and tools'*

Figure 1-1: Thesis statement

1.1. Research motives

Clearly definable problems, such as might be encountered in mathematics, are well-structured (Simon 1973). It is often possible to find a single solution to such problems and then prove that it is correct. Analysis problems are examples of well-structured problems. For instance, a simple hypothesis, e.g. a design can withstand a force X applied at point Y in direction Z, can often be proved correct or incorrect using a closed-world assumption. Not all problems, however, are well-structured. Design synthesis tasks tend to be ill-structured (Simon 1973):

> ***Definition**: An ill-structured problem is a problem whose structure lacks definition in some respect.*

Definition 1-1: Definition of an ill-structured problem (Simon 1973)

Ill-structured problems are open-ended. There may be no prescribed method of solution, no limitation on the number of possible answers and no knowledge of the structure or size of potential design solutions. There is also no guarantee that a solution can be found. 'For the sake of illustration, what might we take as an ill-structured task? Deciding on a career. Discovering a new scientific theory. Evaluating a new ballet. Planning what to do with a free day. Painting a picture. Making conversation with a just re-encountered long time acquaintance. Designing a new house. Making a new invention. Finding a way out of Vietnam. Thinking up a critical experiment to test a scientific hypothesis. Making a silk purse from a sow's ear. Generating this list' (Newell and Simon 1972).

Methods that could assist the solution of ill-structured problems would be of great benefit to engineers. Fully automated design and manufacture has been described as the 'holy grail of the CAD/CAM[2] field' (Lipson and Pollack 2000a). Total automation may be an unattainable goal, but the elevation of the computer from humble workhorse to synthesis partner, capable of working alongside a human designer and contributing innovative and credible design ideas must be considered a possibility for the near future.

The aim of this research is to create and develop computational tools to assist the work of the mechanical systems designer, following on from work already started in this and other fields of research. By attempting to characterise the elements of mechanical systems through parameterised form and function, it is hoped to create examples of how generative methods can be used successfully to aid innovative design of mechanical systems, an important part of being able to address ill-structured problems. 'Creativity is an inspirational force that generates new ideas or produces novel combinations of existing ideas, leading to further solutions or deeper understanding. Creativity is often associated with an intuitive, synthesising approach' (Pahl and Beitz 1996). Without some aspect of innovation or creativity it would be difficult to envisage the production of new ideas. 'Synthesis is not a matter of sheer inspiration. Creativity involves reasoning and calculation, too' (French 1994). Design synthesis problems, though ill-structured, are by no means impossible to solve, though Wielinga and Schreiber (1997) note that 'solving even the simplest forms of configuration design and layout design requires advanced AI technology'. Computational synthesis methods could be of assistance in their solution, acting in concert with a designer to encourage lateral thinking and rapidly generate novel solutions.

[2] Computer Aided Machining.

1.2. Storyboard

The example in the following storyboard is based on an actual design project that was part of the inspiration for this research. The 6-speed gearbox shown in Figure 1-2 was designed for the Audi R8 sportscar that swept to a 1-2-3 podium finish in the 2000 Le Mans 24 hour race, subsequently also winning the 2001 and 2002 events. A classic example of high quality engineering providing a clear-cut technological advantage, it is this kind of design problem that provides the motivation for perfecting computational synthesis tools that can have an impact on future design challenges.

1.2.1. Storyboard scenario

A team of designers have been given the task of producing a bespoke transmission system for a particular car manufacturer. The manufacturer is hoping to enter a race version of its car into a well-known endurance race in the hope of promoting its reputation in the field of quality engineering (Bamsey et al. 2001). A graphical representation of the resulting transmission system is provided in Figure 1-2.

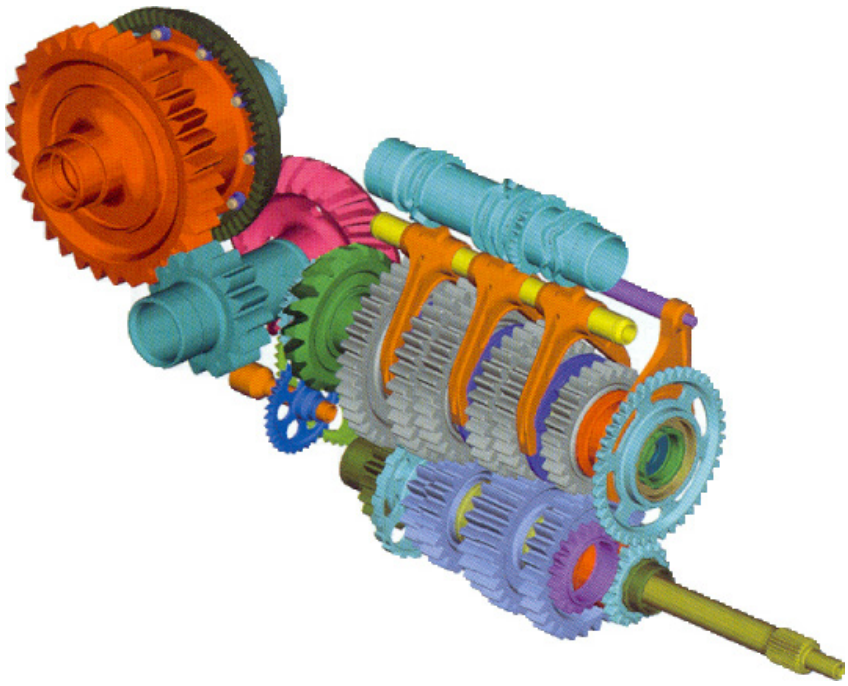

Figure 1-2: Audi R8 Transmission by Ricardo (Bamsey et al. 2001)

The car is already being raced with an older transmission system that is deemed to be out of date. The new system is to fit into the old car, so rough size constraints and power input/output envelopes are already known, though the engine of the new car is likely to be developed further. The main problem with the old transmission is its reliability: the endurance race is tough on transmissions due to a rough track surface and so the designers have been charged with two main targets, (1) to ensure high performance with exceptional reliability, so that there is a good chance that the transmission need not be changed during the course of a race, and (2) to allow an easy change of the complete transmission system in a pit-stop should this be necessary. The former target takes precedence as any unnecessary pit-stop could be the difference between winning and losing the race.

The design team would like to develop a new solution to this design problem, i.e. more than minor modifications to the old transmission system. They would also like to explore as many possible solutions as possible without necessarily going through the whole design and prototype cycle as the race is an annual one and therefore there is limited time for this project.

The team are able to use software that has been designed in-house over the last years, allowing them to take advantage of code libraries of commonly used parts that are required for the new transmission system. The new design ideas that they have for this problem can be added to these libraries, which model function and parametric form. Design proceeds using synthesis tools that permit both manual and computer-controlled exploration of design alternatives, adapting possible embodiments to spatial and behavioural constraints. These constraints include information on manufacturing detail that have been ascertained by the designers over the course of the design process, as well as generic common sense laws such as ensuring no overlap of component parts. The search for good designs incorporates computational optimisation to fine-tune designs, with objective functions targeting the desired solution space, e.g. good endurance characteristics. Of the different results produced, all of which exist as virtual prototypes, the most promising designs are selected for further development, the final choice of design being made by the design team.

1.2.2. Conclusions from storyboard scenario

Consideration of the storyboard scenario highlights some of the key facets required by a successful computational method.

These include:

- parametric design representations that allow movement between different levels of design detail,
- generative methods to create and modify designs at both a functional and spatial level,
- methods of computationally exploring design spaces defined by the generative method to mediate between possible designs,
- performance-based computational evaluation methods to afford fast and accurate feedback on the quality of designs and
- effective data presentation and virtual prototyping for rapid dissemination of synthesis results.

1.3. Thesis structure

This thesis is organised into chapters as follows:

Chapter 1 Introduction: the concept of design synthesis is introduced in the light of the design motives for this research. A storyboard design synthesis scenario, the inspiration for much of this research, is outlined.

Chapter 2 Method: the parametric synthesis framework used for this research is described and placed in the context of existing general design methodologies.

Chapter 3 Background: previous research pertinent to this thesis is discussed and analysed. Particular research efforts are compared and the contributions of this thesis are mapped out with reference to previous work.

Chapter 4 Design generation: a parallel grammar, consisting of function and structure grammars, is introduced as a production system for synthesis of mechanical designs. A clock grammar is considered as a case study for the parametric synthesis of designs composed of gears, spindles and plates.

Chapter 5 Finding preferred solutions: a method of perturbing generated designs is introduced. This extension to the parallel grammar enables design modification, thereby facilitating search for optimally directed designs using simple geometry-based evaluation metrics. Preferred clock designs are generated using the parallel grammar.

Chapter 6 Enhancing design evaluation: a performance-based evaluation method is considered for use in conjunction with the parallel grammar. A simulation model is generated to evaluate designs 'on the fly'. A multi-objective hybrid pattern search method is used to find preferred designs for a camera redesign case study.

Chapter 7 Industrial applicability: the parallel grammar is used for two industrial case studies, the redesign of power drills and vehicle gearboxes. The grammar is modified to allow parallel power paths for these systems, i.e. different output speeds as selected by a user. A modification grammar is introduced to consider alternative designs at a high level of abstraction.

Chapter 8 Discussion and conclusions: the thesis contents are discussed, future work is mapped out and the thesis conclusions are made.

2. Method

A common methodology for engineering design research is shown in Figure 2-1 (Blessing et al. 1998). The four parts provide general guidelines for systematically approaching engineering design research, namely (1) definition of success criteria, (2) description of current design practice, (3) prescription of improvements to current practice and (4) validation of newly changed practice.

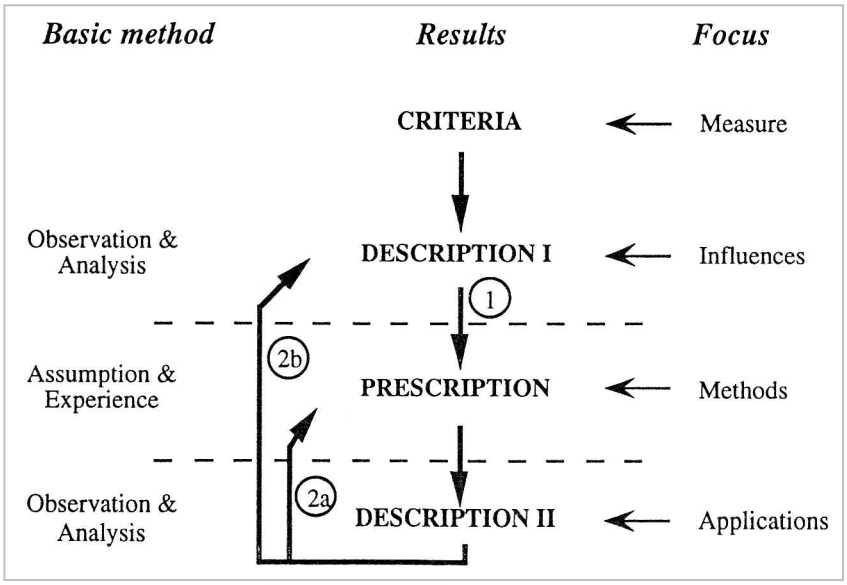

Figure 2-1: Design Research Methodology Framework (Blessing et al. 1998)

Recently, more specific work has been carried out to formulate a methodology for computer-aided engineering design (CaeD). Computer tools are becoming increasingly vital to engineering design. Rapid and virtual prototyping, communication between different software platforms, choices of design representation and computer languages are of great consequence to the success of CaeD. As there is no direct corollary to 'pencil and paper' design, computer-aided design research

requires a more tailor-made approach. A methodology, based on the framework in Figure 2-1, is shown in Figure 2-2 (Bracewell et al. 2001).

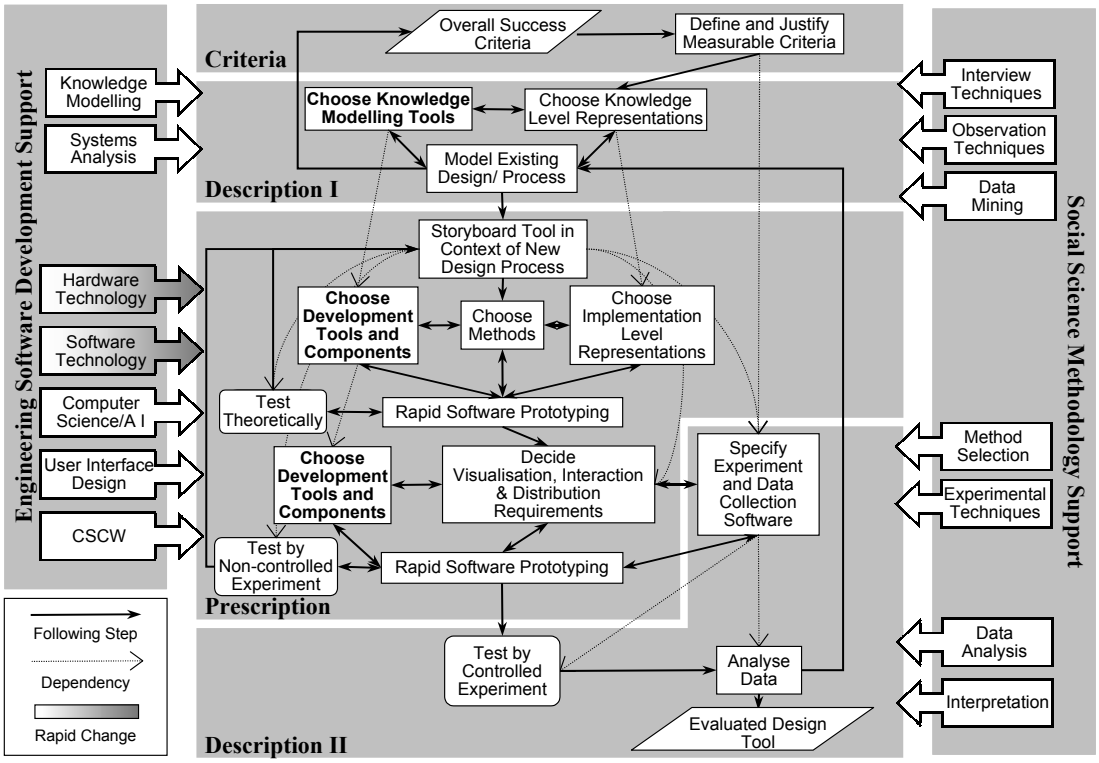

Figure 2-2: Methodology for researching computer aided engineering design tools (Bracewell et al. 2001)

This graphic is very complex and it is outside the scope of this chapter to discuss all its aspects in detail. However, it is a useful summary of the large number of CaeD issues that must be taken into consideration when undertaking software development. This methodology has been validated by implementation of CaeDRe, a **c**omputer-**a**ided **e**ngineering **d**esign **r**esearch **e**nvironment (Bracewell and Shea 2001).

Continuing with the four-phase blueprint, a framework for design synthesis research has been proposed (Starling and Shea 2002) based on a review of general approaches (Antonsson and Cagan 2001) and previous, extensive work in the area of structural synthesis, e.g. (Shea and Cagan 1997). This framework is shown in Figure 2-3.

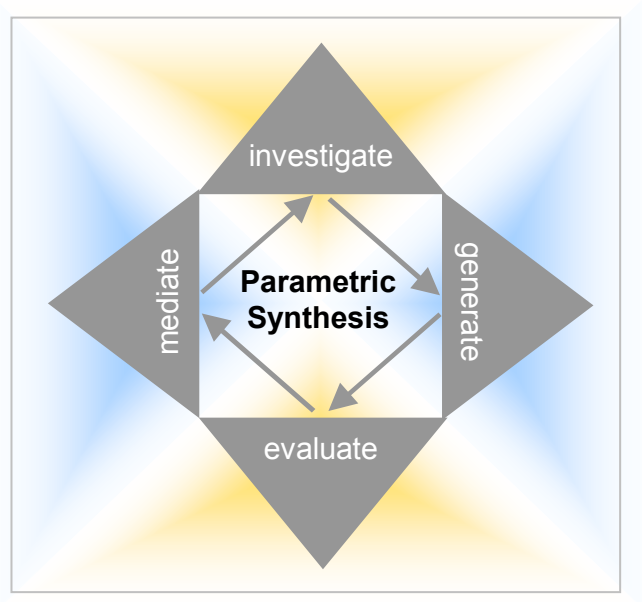

Figure 2-3: Parametric synthesis framework

The phases of the parametric synthesis framework provide a structure for design synthesis research to *investigate* existing design domains, *generate* existing and novel designs using performance-based *evaluation* to *mediate* between preferred novel design alternatives (Shea and Starling 2003). The aim is to provide:

- a bottom-up synthesis model,
- three-dimensional parametric representations of fundamental design components and their direct manipulation,
- sufficient knowledge within design representations to allow quantitative performance analysis using integrated software where possible and enabling possibilities for rapid prototyping,
- simple, flexible, robust and efficient representations of the design domain,
- generation of both innovative and conventional designs, the latter of which can be used for theoretical verification of the system, and

- development of tools that enhance the capability of the designer for innovation and creativity.

Subsequent research has provided analogous frameworks with similar aims, e.g. (Campbell and Rai 2003). Such methods provide a rigorous yet flexible approach to engineering design synthesis and specifically the development of useful computer tools based on these synthesis principles.

The parametric synthesis methodology was followed during the undertaking of the research in this thesis.

3. Background

This chapter places the design synthesis research contributions of this work in the context of existing literature in relevant research areas. The following sections outline the background that is relevant to this thesis. The chapter is structured as follows:

- Section 3.1 defines the concept of design and looks more closely at the nature of the design process.

- Section 3.2 introduces the concept of engineering design synthesis. Relevant work is presented to show the current state-of-the-art in this field.

- Production systems, a formalism for generation of designs, are introduced in section 3.3. Design grammars, a type of production system, are presented as they are particularly applicable to design synthesis research.

- Section 3.4 investigates the concept of a design, i.e. the tangible deliverables of the design process.

- Section 3.5 shows how analysis and evaluation are vital parts of the synthesis process. Behavioural evaluation is introduced as the basis of performance-based synthesis.

- Section 3.6 looks at some of the tools available for search and optimisation that can be used as part of synthesis methods to locate good design solutions. Deterministic and non-deterministic (i.e. heuristic) optimisation methods are

differentiated. Constraints are discussed to highlight their importance for successful optimisation routines.

- Section 3.7 compares selected pieces of existing design synthesis work in detail.

- Section 3.8 summarises the contributions of this thesis with respected to existing research.

3.1. Engineering design

Due to its very nature, attempting a precise definition of engineering design is in itself an ill-structured problem (Simon 1973). There exist a plethora of definitions, for example (Andersson 2001; French 1999; Suh 2001; Yan 1998), that attempt, not without success, to distil its essence. Dym (1994) proposes the following:

> ***Definition**: Engineering design is the systematic, intelligent generation and evaluation of specifications for artefacts whose form and function achieve stated objectives and satisfy specified constraints.*

Definition 3-1: Definition of engineering design (Dym 1994)

Definition 3-1 is useful for the purposes of this work as it clearly outlines the fundamental issues that must be addressed when undertaking a design project. Firstly, it enables the consideration of engineering design as a systematic process. Given an unlimited amount of time, one can imagine an infinite number of proverbial monkeys producing the complete works of Shakespeare (Elmo et al. 2002). Practically, however, infinite time is not an available resource. Hence the need for 'systematic, intelligent generation and evaluation'.

Secondly, this definition targets the creation of real objects, or artefacts, that are characterised by their 'form and function', in contrast to pure spatial design as might be considered in the visual arts and architecture. Design representation issues for computational design are addressed in section 3.4.

Thirdly, and perhaps most importantly, Dym's definition highlights the importance of design activities '[achieving] stated objectives'. While this statement may give the impression of stating the obvious, achieving a number of objectives is not always an easy task as they may, and indeed often do, come into conflict. For example, a new power drill might be required to have a high degree of safety and also be inexpensive. Finding the correct trade-off between these two fairly simple objectives will not always be a straightforward task. Nevertheless, it is the responsibility of the design engineer to develop solutions that provide good compromises to such problems.

Finally, Dym's definition emphasises the need to 'satisfy specified constraints'. It would be wrong to perceive constraints solely as a straitjacket on design. Instead, constraints provide an opportunity to design with a degree of accuracy, to create a product that precisely fulfils the design requirements. Examples of constraints are behavioural requirements, aesthetics, environmental and safety issues, price and ergonomics. Constraints can be classified in many ways, for example hard or soft, wish or must, achievable or not, equality or inequality. Interpreting design needs and specifying the correct form to model constraints is a very important part of the design process.

Understanding the design process goes further than giving a clear definition, as successful design remains a hazy concept even for those who would claim to understand it well. An often-quoted breakdown of the design process is given by Pahl and Beitz, who divide it into four stages, (1) planning and task clarification, (2) conceptual design, (3) embodiment design and (4) detail design (Pahl and Beitz 1996). Though these phases are sequentially arranged, Pahl and Beitz recognise the need to allow a certain degree of overlap between them as well as iteration. The flowchart in Figure 3-1 shows an overview of the general planning and design process.

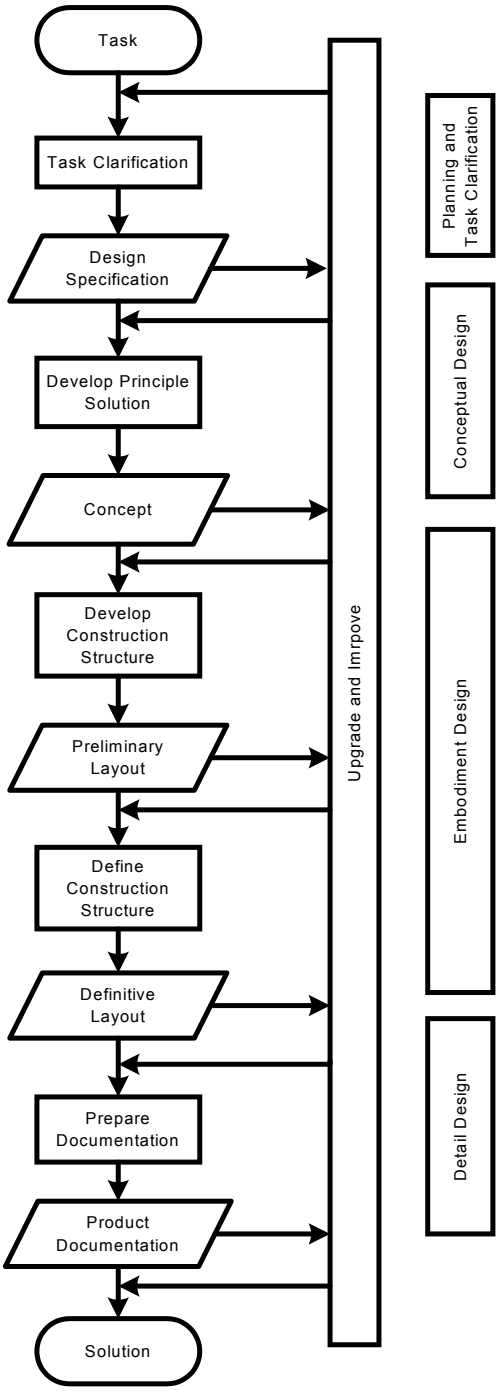

Figure 3-1: Steps of the planning and design process (Pahl and Beitz 1996)

Ulrich and Eppinger (1995) outline a generic development process focussed specifically on product design, encompassing marketing and manufacturing as well as design. Their process is made up of five phases, of which the first four are very similar to the Pahl and Beitz design stages. These are (1) concept development, (2)

system-level design, (3) detail design, (4) testing and refinement and (5) production ramp-up (Figure 3-2).

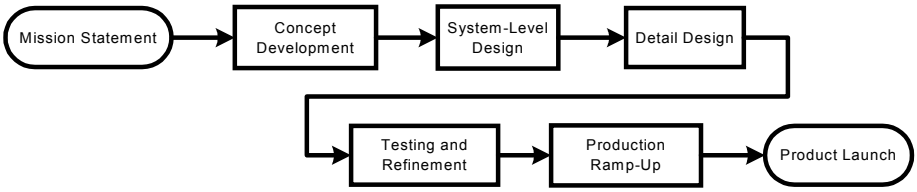

Figure 3-2: Generic development process (Ulrich and Eppinger 1995)

Despite a slight contrast in emphasis between them, there is a large degree of overlap between the planning and design process of Pahl and Beitz and the generic development process of Ulrich and Eppinger. The former is a systematic approach to engineering design in general, whereas the latter is geared more specifically towards product design.

Another discourse on product design is provided by Otto and Wood (2001). With a focus on reverse engineering and redesign for new product development, the authors outline three main steps of a typical development process: (1) understand the opportunity, (2) develop a concept and (3) implement a concept. In the case of product development based on redesign, the design stages are slightly altered. A graphical representation of Otto and Wood's product development process is shown in Figure 3-3.

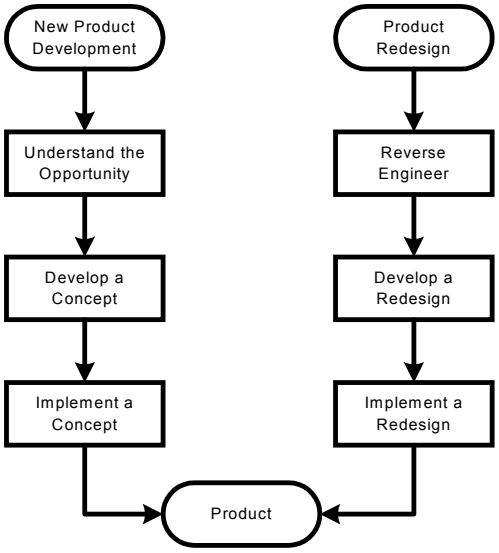

Figure 3-3: Product development process (Otto and Wood 2001)

The clear demarcation between design phases shown in these diagrams is for clarity of representation. In practice, some design tasks bridge these divisions and their classification into one particular design phase is not possible. Iteration is an important part of all design process models. Apart from the three mentioned here, there are other models that have particular emphases, such as the brand-recognition side of product creation and placement, e.g. (Cagan and Vogel 2002). From an engineering standpoint, the three models introduced here suffice as they give a firm basis from which to research computational synthesis. Where possible, Pahl and Beitz will be used as the main reference for design process terminology to avoid confusion arising from the use of different nomenclature.

A final point to bear in mind is that while these models are meant to be descriptive, i.e. illustrating how successful engineering design processes have worked in the past, as well as prescriptive, i.e. outlining how successful engineering design processes can be managed in the future, they are by no means exclusive. A solid engineering process model, followed to the letter, is no guarantee of success. Equally, many designs have been, and will continue to be, successfully produced without rigid adherence to the principles behind the models introduced above.

3.2. Engineering design synthesis

Synthesis research, in the form of computational production systems, can be traced back to the advent of the computer (Post 1943). Synthesis is a fundamental task of engineering and can be thought of as attempting to create form to fulfil desired behaviour (i.e. function). Existing research can be differentiated into computational and non-computational, i.e. 'paper-based', methods.

Raphael and Smith (2003) describe synthesis from a logical point of view as being the antithesis of analysis:

> ***Definition**: Synthesis is the reverse of analysis, where target behaviour is used to infer a physical configuration within an environment.*

Definition 3-2: Definition of synthesis (Raphael and Smith 2003)

This definition does not capture the concept of generating designs as a formalism within the parametric synthesis framework (Figure 2-3). Antonsson and Cagan (2001) provide a more applicable definition of engineering design synthesis:

> ***Definition**: Synthesis is the creative step itself: the conception and postulation of possibly new solutions to solve a problem.*

Definition 3-3: Engineering design synthesis (Antonsson and Cagan 2001)

Design synthesis is commonly a manual task: 'In most engineering design, this step is performed by creative human minds'[3]. Engineering design synthesis can be referred to as being *formal* engineering design synthesis when it is 'computable, structured and rigorous, not *ad hoc*'[3]. The aim of this research is not to develop computational creativity, but aid the creativity of designers through the provision of formal synthesis methods for mechanical design, i.e. to enable the generation of novel mechanical design solutions using computer tools.

[3] These quotations are both obtained from (Antonsson and Cagan 2001).

Design synthesis of mechanical systems is a non-trivial task. Whitney (1996)[4] compares VLSI (Very Large Scale Integration – referring exclusively to transistors) and complex electro-mechanical (CEM) design to explain why the computational generation of mechanical designs is a hard problem. Four specific reasons are outlined in Table 3-1 that complicate the synthesis tasks for CEM design.

Table 3-1: VLSI versus CEM design (Whitney 1996)

	Main distinctions	**Rationalisation**
1.	Mechanical systems carry significant power at high tolerances	Multiple side effects of high power levels are not of particular issue in VLSI design. CEM designers spend a large proportion of their time 'anticipating and mitigating a wide array of side effects'.
2.	VLSI systems are predominantly signal processors	VLSI circuit design can be building-block based without altering system behaviour.
3.	Single versus multiple functions per device	The function of VLSI elements is logic; other functions are usually negligible. Function sharing (Ulrich and Seering 1990) is more prevalent in CEM and functions are not usually logic-based. Intricate reasoning strategies are necessitated by back loading[5], the interaction of components in a design at run-time.
4.	Ability or inability to separate component design from system design	Due to back loading, system and component design are inseparable in CEM design. For example, particular components might require redesign due to possible system configuration. This results in further complication.

Essentially, these differences stem from the significant levels of power transmission inherent in mechanical designs. Burgess et al. (1997) have evidence of significant scale effects in mechanical engineering design, using flexible hinge elements as an example. Ward (2001) argues for the need for mechanical design compilers, i.e.

[4] At the time, the publication of this paper provoked excitement in the design community, Antonsson strongly arguing that Whitney was being too pessimistic about the potential for automation of mechanical design tasks (Antonsson 1997). After further correspondence (Antonsson and Whitney 1997) the two protagonists agreed to couch their arguments in terms of challenging research agendas.

software for formal engineering design synthesis. These compilers must be able to (1) provide a flexible high-level language to define mechanical systems, (2) accept any syntactically correct and semantically meaningful input which is expressed in this language and (3) transform this input to a manufacturing system to produce high-quality designs. Success of these mechanical design compilers hinges on being able to overcome the inherent difficulties associated with mechanical design synthesis. The aim is to be able to generate innovative design solutions, producing mechanical structures, ideally from first principles (Cagan and Agogino 1987).

Hoover and Rinderle (1989) propose a graph-based strategy for design of mechanical devices to incorporate functional integration and take incidental behaviour into account. This paper-based method is not able to produce strongly innovative designs as design make-up is limited to fixed sets of non-parametric component libraries. An abstract description of the desired behaviour is put together as a specification graph. To obtain a representation of a design that can be implemented, function-preserving transformations are applied so that the graph more closely resembles a collection of actual, or theoretically possible, components. Transformations such as these that do not alter functionality are referred to as being function-preserving. e.g. the transformation of a 9-1 gear pair into two sequential 3-1 gear pairs. On a more detailed level these changes result in different behaviours, as using two pairs rather than one will change the direction of rotation as well as introduce spatial consequences.

The resulting design description is then transformed further by carrying out component-directed transforms. Relevant components are located in a database and compared with the transformed graph for validity, e.g. ensuring that usability ranges of inputs or outputs are not compromised. If the component is suitable a component-directed transform is invoked to replace the portion of the graph under consideration with a description of the chosen part.

[5] Attaching a module to a power train, such as a load to a motor, results in a back effect that alters the behaviour of the existing power train, i.e. in this case the motor will start drawing increased levels of power. This is referred to as ‘back loading’.

Ulrich and Seering (1989) present a paper-based method for schematic synthesis of devices that can be represented as networks of lumped-parameter systems. Kota and Chiou (1992) have investigated the computational synthesis of mechanisms using functional and physical building blocks. A qualitative matrix representation is used to describe building blocks. A motion transformation matrix (MTM) holds information on input and output rotations and translations with respect to Cartesian co-ordinates. Constraint matrices holds information such as the reversibility (or otherwise) of allowed couplings and behaviour such as intermittent motion. By using row shift, column shift and decomposition operators, these matrices can be manipulated in order to find sub-function MTMs that match those of existing building blocks. These building blocks can then be used to synthesise new solutions to the problem represented by the original MTM.

Further work introduces a lower-level representation to provide a geometric relationship between input and output axes of building blocks (Chiou and Kota 1999). The implementation includes a library of 43 building blocks representing different elements. More recently, Moon and Kota (2002) use dual vector algebra for transformation and alignment of building blocks as part of topological synthesis.

Campbell et al. (1999; 2000; 2003) propose agent-based design (A-Design) of electro-mechanical systems. A hierarchy of software 'agents' work semi-autonomously to produce a number of designs based on input from a user, who has the opportunity of controlling the process by indicating when to choose certain design goals. A-Design was demonstrated to work effectively in electro-mechanical configuration design for a weighing machine design case study (Campbell et al. 1999).

3.3. Production systems

Post (1943) first coined the term production system as a general but formal model of computation. A number of different production system formalisms have been applied in different research areas, including linguistics, programming, architecture and spatial design (Gips and Stiny 1980). Knowledge-based systems (KBSs), often referred to as expert systems, are an example of production systems. The 'expert'

comes from the fact that knowledge-based systems use information from experts and often achieve expert-level performance in design (Stahovich 2001). There are many KBS definitions, e.g. (Dym and Levitt 1991; Sriram 1997) and references to further sources cited by these authors. A useful definition of a knowledge-based system is given by Dym and Levitt (1991):

> ***Definition**: A knowledge-based (expert) system is a computer program that performs a task normally done by an expert or consultant and which, in so doing, uses captured, heuristic knowledge.*

Definition 3-4: Definition of a knowledge-based (expert) system (Dym and Levitt 1991).

Knowledge-based systems have been shown to successfully solve complex and ill-structured problems (Sriram 1997) but are less suited to geometric reasoning (Stahovich 2001) due to the difficulties in representing geometry with a small set of simple axioms (Forbus et al. 1991).

Lipson and Pollack (2000a; 2000b) have implemented a sophisticated production system for design synthesis. Using genetic algorithms, they are able to generate robots out of struts and actuators using a fitness function based on locomotive ability. A rule-based synthesis strategy is used to modify parametric bars, ball joints and actuators.

Chakrabarti and Bligh (1996; 2001) and Chakrabarti et al. (2002) use a 'compositional synthesis' approach to generate solution concepts with different levels of abstraction to address coupled mechanical design problems with their functional synthesiser 'FuncSION'. Exhaustive search is employed to generate vector-representations of potential solution concepts of single-input, single-output problems with orthogonality restrictions. A hierarchy of representation levels are used to represent full solutions: (1) a topological solution, (2) a spatial configuration and (3) a physical solution representing form. The spatial configuration consists of vectors that represent functional elements such as tie rods, screws, levers, wedges and cams. Combination of these elements allows solution representation with basic topology as prescribed by

the top level of the representation hierarchy. The form representation has neither been fully implemented nor linked to the main generation software.

Liu et al. (1999) have improved FuncSION to include spatial configurations, using heuristics to 'prune' infeasible solutions. Compatibility between elements is maintained by ensuring that normal vectors of contact points are coincident and both form and motion requirements are fulfilled, that is, pieces fit together and required movement is not restricted. This work also exhibits computer-aided sketching of generic three-dimensional embodiments, though this has only been implemented for two simple components, a wedge and a lever.

Wahl et al. (2003) have developed a general framework for synthesis of three-dimensional one degree-of-freedom mechanism models. Intended for the early phases of conceptual design, this assistance tool uses two synthesis steps to generate layout information of potential designs from a pre-defined library of elementary mechanical blocks. A series of solutions are generated using qualitative reasoning and, after further analysis, are presented to the designer.

3.3.1. Design grammars

Chomsky (1957) was the first to use the word 'grammar' in the technical context of production systems (Post 1943) while developing string grammars to generate valid, i.e. 'grammatically correct', linguistic sentences. Other grammars have since been developed, e.g. shape grammars (Gips and Stiny 1980). The use of grammars to assist design is conceptually simple. In the same way as a natural language is based on rules (termed the grammar), it is possible to develop a language of designs. Starting with a legal construct, repeated application of different grammar rules generates new designs. The sum of different designs produced by application of grammar rules in this manner is termed the design language.

Design grammars lend themselves to computational implementation in much the same way as scripting languages are a useful tool for writing computer code. High level scripting languages are powerful, yet flexible and easy to use. Writing a piece of code

in a scripting language to carry out a series of computing tasks might take longer than carrying out those tasks manually. Once written, however, the script can be used many times to reproduce different variations of that series of tasks much more efficiently than one could ever achieve by hand.

Similarly, once a grammar has been written, it can be employed repeatedly to discover new design possibilities. Paper-based application of grammar rules is of limited use, but a computational grammar implementation holds increased potential for the exploration design spaces in a more thorough manner than could be achieved manually. There is a large body of existing research investigating of the use of design grammars. Some examples include little or no implementation work, concentrating on developing the grammars themselves, while others have put considerable effort into creating usable prototypes, using optimisation algorithms to find designs.

Schmidt and Cagan (1997) highlight three main reasons why grammars are useful for mechanical engineering design: they can (1) express and retrieve any possible design from a full space of design possibilities, (2) transform the design process into a direct computational procedure and (3) recognise emergent mechanical behaviour. Regarding point (1) it must be noted that designs produced by a grammar are limited to the language of that particular grammar. A grammar-based design system will be unable to produce new solutions outside the scope of its language. This is not so much a limitation on grammar usage but rather should be viewed as a challenge to formulate flexible and powerful grammar rules.

Point (3) above introduces the concept of emergence. This phenomenon occurs when large collections of very simple systems exhibit complex macro-behaviour. For example, it is believed that nesting ants interact in a bottom-up manner – a few simple rules governing individual ant behaviour result in complex, so-called emergent, behaviour of the whole ant colony (Johnson 2001). Emergent behaviour can be beneficial to a design but can also introduce unwanted side effects (Chakrabarti and Johnson 1999). Hence due to the high propensity for mechanical designs to incorporate some sort of function-sharing, an understanding of emergent behaviour is important for grammatical design, a bottom-up process (Brown 1997; Stiny 1991).

3.3.1.1. Shape grammars

George Stiny (1980b) introduces the concept of a language of design using shape grammars. His work is based on the 'kindergarten method' of Frederick Froebel – a method of teaching young children about spatial awareness using 'constructive play'. Children's play blocks are introduced to a child in stages of increasing complication to encourage the child to use the blocks available to create forms. Languages of design are based on a vocabulary of blocks that are available for use. It is then possible to define spatial relations that are legal, which leads on to the formulation of shape rules, for example the construction of an arch using specific building blocks. Starting from an initial shape, a complete grammar of such rules can be used to generate constructions that aim to have characteristics of three different categories: forms of knowledge, life and beauty. In relation to optimisation, this basic analysis already introduces the concept of an objective function: the goal being to attain solutions that satisfy, or satisfice[6], the conditions laid out in the different categories.

Stiny (1980a; 1991) develops the idea of shapes up to and including three dimensions as the basis for algebras that form the objects of shape grammars. Including labels in an algebra allows more information to be conveyed in the shape representation. Rules can be defined that act on particular shapes and label combinations. This allows for flexibility in defining rules that are relevant to particular situations, rather than just looking for shape considerations. Similarly, weights can be added to shape grammars to enrich design properties (Stiny 1992), for example a line can be given a weighting that describes its thickness. The idea of weightings can be extended, for example by assigning different radii to points, different shadings to planes and literally different weights to solids.

Shape grammars can be formally defined thus. Rules are of the form $A \rightarrow B$, where both A and B are shapes. If the shape in the left hand side (LHS) of the rule, i.e. A, can be found in an arbitrary design C then this rule is a valid potential rule application. Having determined the applicability of a rule, i.e. the presence of A

[6] To satisfice means to 'decide on and pursue a course of action that will satisfy the *minimum* requirements necessary to achieve a particular goal', i.e. without necessarily fulfilling the *desired*

within C, the designer or algorithm-controlled decision-maker can choose whether or not to modify the design C by applying the rule. If it is chosen to proceed, the transformation $C \rightarrow (C - t(A)) + t(B)$ is carried out, replacing the LHS (A) in shape C by the right hand side (RHS) shape (B), where t is the transformation that makes A a sub-shape of C. The language of a grammar consists of the members of the solution set, i.e. all possible results C that can be produced from a specified starting shape S after application of different sequences of the possible rules in the grammar.

A grammar is defined by a 4-tuple $G = (V, X, R, S)$, where V is the set of objects that are manipulated by the grammar, X is a set of terminal and non-terminal symbols, S is the initial symbol and R is a set of rules of the form outlined above. The language of the grammar G is the set of all results produced from the start symbol that consists of only terminal symbols. Having terminal and non-terminal symbols is meant as a flag to indicate whether or not a design is complete. Hence formal grammars will have specific rules whose sole purpose is to replace non-terminal shapes and labels with equivalent terminal symbols to complete the design.

Early design work using shape grammars can be found in the field of architecture (Knight 1994). A recent example is mass customisation of affordable housing. A house grammar has been shown to be a useful tool for elucidating the language of possible templates for designs of a particular style, specifically Alvaro Siza's houses at Malagueira (Duarte 2003; Duarte in progress). Chase (1996) has worked on using shape grammars with symbolic logic to produce tools capable of manipulating shapes while upholding geometrical, topological and logical constraints. This work has found applications in geographic information systems and architectural plans.

Research into engineering design grammars has also resulted in advances in structural design generation. Shea and Cagan (1998) have developed a truss structure grammar that was used to generate novel designs during a study of the roof trusses of a building at Carnegie Mellon University. This grammar was used in conjunction with a shape annealing algorithm (Cagan and Mitchell 1993) and different design constraints to

---

requirements. This is meant in contrast to a similar verb, to satisfy, that indicates providing implied total satisfaction or contentment (source: Oxford English Dictionary).

develop optimally directed[7] designs. The designs produced for the roof truss were found to compare favourably with separate designs produced by architects and structural engineers who proposed solutions for the same design task. Other examples of this work include innovative dome designs (Shea and Cagan 1997) and transmission tower redesign (Shea and Smith 1999). The structure grammar is now implemented as a general truss design tool called eifForm (Shea et al. 2003). Similar work has been carried out for bicycle frame design (Suppapitnarm et al. 1999). A multi-objective simulated annealing (MOSA) algorithm was used to redesign diamond frame configurations that agree with common wisdom on bicycle design.

Much effort has been invested into developing grammars for mechanical designs and examples exist for form generation of diverse products such as Harley-Davidson motorcycles (Pugliese and Cagan 2002) and coffeemakers (Agarwal 1999). Earl (1987) discusses the use of shape grammars for generation of form as well as function. Finger and Rinderle (1989; 2002) have developed a grammar based on the manual manipulation of form-behaviour diagrams using bond graphs (Paynter 1961) for conceptual design of topological configurations from a part-based element library. Geometry is considered without generation of detailed form. A series of different manual transformations (behaviour-preserving, component-directed and other) are used to produce different designs. This research is one of the early pieces of work in mechanical design grammars and has not been implemented computationally, due to a lack of a software interpreter for parametric shape grammars.

McCormack and Cagan (2000) have developed a two-dimensional parametric shape grammar interpreter and demonstrated its use for the design of vehicle inner bonnet panels. This inner hood[8] grammar has been used to generate new bonnet designs using agent-based design optimisation. The shape grammar interpreter follows a step-wise classification process that enables the interpreter to decide which design sub-shapes particular rules should be applied to. The process differentiates between different sub-

[7] When seeking solutions to ill-structured problems it can be impossible to tell with certainty whether the best solution found so far is a global optimum or not. In cases like this, such solutions can be referred to as being optimally directed (Cagan and Mitchell 1993). See also section 3.6.
[8] 'Hood' is an American expression for the English term 'bonnet' (source: Oxford English Dictionary).

shape groups by calculating line intersections and considering line symmetry. This inner hood grammar is limited to shapes consisting of straight lines.

A shape grammar interpreter that can compute shapes with curved lines would enhance the possibilities for producing interesting designs. An example of such a 'curvy' grammar is the parametric Buick grammar (McCormack et al. 2002). The grammar represents a two-dimensional view of a vehicle from a front-on perspective. Modified rule sets are used to examine the Buick automobile brand. A curved line interpreter is under development that uses a two step approach to (1) perform shape matching with equivalent straight-line shapes and (2) verify transformations on the actual curved lines (McCormack and Cagan 2003). As an example, the shape interpreter has been used in conjunction with the Buick grammar to generate an alternative headlight design for a Buick Rendezvous[9].

A shape grammar for mechanical systems will have to be able to describe curved shapes in three dimensions. Work has also been done on boundary solid grammars, considering complex solids and operations that can be carried out on them, such as merging and inverting (Heisserman and Woodbury 1994). Other work analyses special shape grammars that can be used to recognise shape boundaries (Krstic 2001).

Agarwal and Cagan (2000) have used grammars to generate Micro Electro-Mechanical Systems (MEMS) resonators. Using a parametric grammar with weights and labels, it is possible to generate different MEMS that fulfil minimum conditions for legality of such structures, e.g. each generated MEMS resonator has at least one spring and one actuator. Function and form are represented in the MEMS grammar through the use of appropriate labels.

Boeing use a computer-aided design system called Genesis for grammar-like generation of aeroplane designs (Heisserman and Mattikalli 1998). Due to the functional demands of the innately mechanical systems of an aeroplane, the developers of Genesis have had to contend with the abstraction of function from form,

[9] The 'curvy' grammar interpreter currently exists as a paper-based method. Illustrations of such curved designs are, as yet, still created by hand.

resulting in a potential basis for the development of bottom-up synthesis in mechanical designs, as opposed to a more rigid top-down approach.

Schmidt and Cagan (1995) generate solutions to a power supply generation problem using string grammars. The tool 'FFREADA' (Function to Form Recursive Annealing Design Algorithm) is used to produce designs from a library of machine components and their functional attributes. A hierarchical simulated annealing method is used to search for good designs. The same algorithm is also used to generate optimally directed hand-held power drill designs (Schmidt and Cagan 1998).

3.3.1.2. Graph grammars

FFREADA was superseded by a new algorithm, GGREADA (Graph Grammar Recursive Annealing Design Algorithm), that was developed for tasks that feature non-serial function and form dependencies, including a limited amount of function sharing (Schmidt and Cagan 1997). Grammar rules are used to modify graph-based representations (Al-Hakim et al. 2000) that are hierarchical abstractions of designs, hence these are referred to as graph grammars.

The ability to reason about the capability of particular components to fulfil functional requirements is a prerequisite for being able to transform a design concept into a collection of machine components whose spatial arrangement and connectivity combine to make a viable design. As an example, the case study investigated is the top-down design of a small mechanical cart out of components using Meccano, a toy construction set. A supply of a limited set of components and sub-assemblies is made available, with functional goals set for the design of carts. These functional goals include ensuring that the cart can roll and has a flat space that can support a load. Minimum weight is used as the objective function for the search aspect of the algorithm.

The space of designs produced for a particular problem statement is typically ill-behaved, as small changes to a design can affect its fitness for fulfilling the desired specification. For example, removing a wheel component of a small cart design,

essentially a small change, can result in a cart that does not roll properly, and so does not fulfil the conditions in the design specification that require it to be able to roll.

GGREADA does not recognise emergent behaviour. The authors indicate that a grammar capable of recognising emergent behaviour would require a representation that models all possible behaviours in a standard manner, such as a bond graph representation (Paynter 1961). Challenges leading on from this work include moving away from structural tasks based on fixed part libraries to more open-ended design problems with more degrees of freedom.

Graph grammars have been used to build platforms for product families, where they describe the design envelope of these families (Siddique and Rosen 1999). The example of coffeemakers was taken to show common and optional functions for this particular product family. This research is being expanded to develop a Product Family Reasoning System (PFRS) to represent members of product families (Siddique and Rosen 2001).

Other research has investigated the use of a graph-grammatical approach to synthesis of mechanisms, specifically for epicyclic gear train (EGT) mechanisms (Schmidt et al. 2000). This work has been implemented as a grammar-based designer assistance tool for EGT design (Li and Schmidt 2000). The assistance tool creates functional schematics from graphs, where edges are joints (weights are used to distinguish gear and revolute joints) and vertices are gears and carriers. Exhaustive search is used to generate concepts. While not in itself able to generate novel designs, the tool was able to assist the process of innovation. Following the discovery of a new ring-plate type cycloid drive (RCD) design that was not in the language of the original grammar, a restricting assumption was removed from the EGT grammar by adding an additional grammar rule to expand the language of described designs. This new language incorporates the newly discovered cycloid drive design and the new grammar was used to generate novel RCD-type designs (Li et al. 2001).

Another graph grammar-based approach is the generic formalism '43'[10], a non-platform-specific design framework that aims to provide a ready-to-use implementation for different design domains (Alber and Rudolph 2003). Alber et al. (2002) propose an object-oriented approach to design representation using graphs and cite a space-station configuration problem as an example of the type of design their method is capable of generating.

3.4. Representation and functionality

Having discussed the design process in general and design synthesis in more detail, it is important to define what is meant by a design artefact. What constitutes a design, in particular a mechanical design? Stiny (1990) proposes the following:

> ***Definition**: A design is an element in an n-ary relation among drawings, other kinds of descriptions, and correlative devices as needed.*

Definition 3-5: Definition of a design (Stiny 1990)

This definition is useful as it indicates that a design is essentially a model of the final implemented product, structure or entity that is to be created. Designs must be able to convey all the pertinent information necessary for the representation and subsequent creation of the artefact.

3.4.1. Function, behaviour and structure

Definition 3-5 introduces the concept of some sort of connections, or 'correlative devices'. This concept results from a need to represent more than just the physical characteristics of a design. Interest in a design that is being created will be determined by the purpose that it is supposed to fulfil. How can this *function* of a design be represented, and, most importantly, be linked to the eventual form of the design? Similarly, the predicted and actual *behaviour* of a design implementation are of

[10] The name of this framework derives from a well known work of fiction that gives the answer '42' to the Ultimate Question of Life, the Universe and Everything (Adams 1988). The aim of Alber's

importance. Linking behavioural information to form and function considerations enables a more complete depiction of design intentions and ultimately leads to a more effective representation.

Umeda et al. (1990) have made a formal definition of a Function-Behaviour-Structure (FBS) relation (Figure 3-4). Structure and State are seen as interchangeable terms in this work as it is argued that state is time-dependent structure, so for instantaneous snapshots of an entity the use of these terms are one and the same. Behaviour is linked to structure by physical laws, this is termed the B-S relationship; function is linked to behaviour by a human-specified mapping Γ_{ab}, this is termed the F-B relationship. These connections allow the three parts of a design, i.e. function, behaviour and structure/state, to be linked together to model a particular design. Finally, the context of the FBS model is determined by the view, i.e. the 'physical situations of the current interest' (Umeda et al. 1990). FBS models have been previously used for grammatical design: Chase (2002) outlines a model for redesign based on a Function-Behaviour-Structure model using graph grammars.

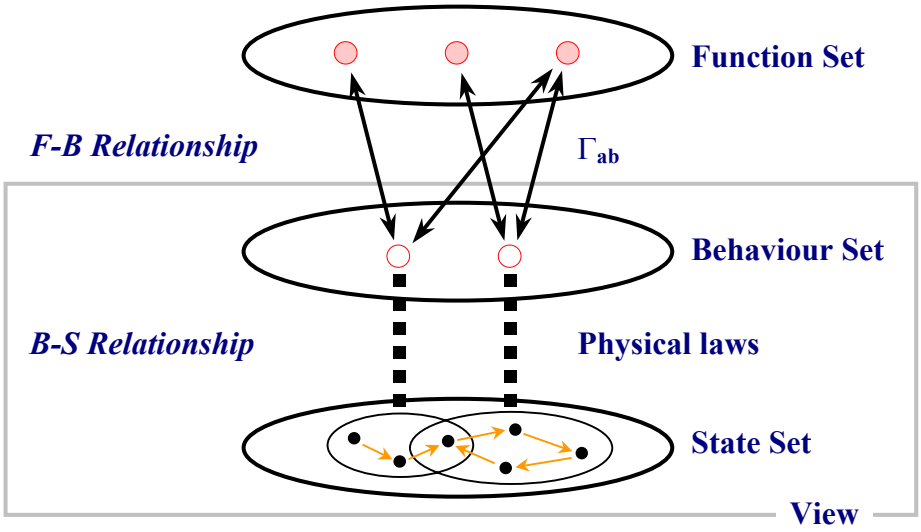

Figure 3-4: Function-Behaviour-State (or Structure) model (Umeda et al. 1990)

Winsor and MacCallum (1994) present a review of functionality modelling in design. There exist different perceptions of functionality: use of the word 'function' as a verb or noun results in different meanings. Used as a noun, 'function' indicates a sense of

approach is to go one better than this, hence 42 + 1 = 43.

purpose, i.e. the over-riding *raison d'être* of an object. Used as a verb, the same word conveys how an object works, i.e. what actual physical processes take place while the object is in use. Hence one might say that the function of a bicycle is to transport people from place to place, but a bicycle functions by using energy from a person's body to produce a torque that acts on the wheels of the bicycle, ultimately resulting in translational motion. To distinguish between the two, it is sensible to differentiate between function (as purpose) and function (as action), the latter being referred to as behaviour. Hence use of the noun should be reserved for the first meaning where possible. Finger and Rinderle (1989) propose a definition of the terms function and behaviour:

> ***Definition**: The function of a design is what it is used for, the behaviour of a design is what it does. Hence function indicates the subset of behaviours that are required for a device to work correctly.*

Definition 3-6: Definition of function and behaviour (Finger and Rinderle 1989)

A successfully designed product fulfils the tasks required of it, i.e. its function, without violating any constraints, such as might be governed by environment, for example limits on height and weight, and usability, for example non-toxicity. Definition 3-6 conforms to usage of the term 'function' in this field of research, e.g. by Chandrasekaran and Josephson (2000), Pahl and Wallace (2002) and Simon (2001). There are different suggested methods for representing functionality. Function grammars are used as a formalisation of common language usage to express function as a verb-noun pair. Function can be represented as input/output relationships, indicating a flow of information, matter and energy (Pahl and Beitz 1996; Ulrich and Eppinger 1995). Functional decomposition of 'black box' models with object/verb pairs allows detailed understanding of sub-functions, however, this reliance on flow can result in difficulties when attempting to classify the function of a fixed structure, such as a shelf supporting books.

Early work in this field includes the Edinburgh Designer System (EDS), developed by Popplestone (1987) to infer modules based on rigid bodies, features[11] and connecting ports. Other related work includes the field of Qualitative Physics, a method of representing functionality using rules of thumb to describe the physical world (De Kleer and Brown 1984; Forbus 1984). Stone and Wood (2000) have developed a functional basis for design to aid systematic function structure generation. Rosen (1993) investigates the concept of feature-based design for CAD systems.

3.4.2. Functional integration

Hoover and Rinderle (1989) have investigated aspects of functionality in mechanical systems. They note how components in good mechanical designs are often highly inter-related: many components fulfil multiple purposes in the complete design, i.e. there is a high degree of function sharing (Definition 3-7). This is because having a 'one-function, one-component' strategy is often not viable, economically or spatially, for mechanical systems, unlike many others design domains (Whitney 1996).

> ***Definition**: Function sharing is the simultaneous implementation of several functions by a single structural element.*

Definition 3-7: Definition of function sharing (Ulrich and Seering 1990)

Furthermore, good mechanical designs make use not only of the intended function of components or sub-assemblies, but also utilise their behaviour in the given circumstance. In parallel with the purpose/action distinction (Winsor and MacCallum 1994), Hoover and Rinderle suggest that the difference between behaviour and function is intent. Behaviour can be utilised deliberately or be incidental, the former indicating that this behaviour contributes towards the fulfilment of the intended function and the latter that it does not. Purely functional components do not exist, as all components will have some sort of behavioural data associated with them. Function and behaviour are therefore irrevocably linked and must be considered

[11] A feature is an interpretation of geometry and conveys meaning about components beyond that supplied by geometry alone (Brown et al. 1995).

simultaneously. A design that fulfils functional requirements adequately can only be considered a favourable design after analysis of its behaviour.

For example, the function of a casing around a machine might be to fulfil Health & Safety requirements in order that a user cannot damage his or her fingers. The behaviour of the casing might also contribute to the structural integrity of the whole machine. A good design will ensure that this behaviour is taken into account. An opportunistic rather than compensatory approach to this incidental behaviour, which can also be emergent behaviour, will result in better designs.

Jensen (2000) portrays functional integration through the use of 'wirk' elements, based on domain theory (Hansen and Andreasen 2002). These are derived from the German noun *Wirkung* which defies easy translation into English[12]. Wirk elements can be used to describe and analyse functional integration. Design models consist of wirk elements, termed the organ domain, and form elements, termed the part domain. Structural and behavioural aspects of mechanical artefacts are linked using a function-means tree (Hansen and Andreasen 2002).

Hoover and Rinderle (1989) use an approach that relies on abstract primitives with component-directed transforms to move towards a final design. Ulrich and Seering have looked at ways of using 'design and debug' methods to synthesise designs by using different computational modules called 'perspectives' for different design goals (Ulrich and Seering 1987). This has been demonstrated for mathematical control tasks as well as for the design of a car speedometer.

Genesis (Callahan and Heisserman 1997; Heisserman and Mattikalli 1998) uses a hierarchical assembly representation that can describe complex product designs. The assembly representation allows each physical part to be included in a design, while minimising any necessary repetitions in data storage. The representation used by Genesis is therefore suitable for large assemblies, such as aircraft design. Components that are used in large numbers, for example a rivet or a bolt, have a separate data store for information that is the same for all instantiations. However, instantiation-specific

[12] A loose translation of the German word *Wirkung* might yield something like 'active effect'.

data (such as a protective covering required in an area subject to corrosive substances) gets stored separately in the part-occurrence area of the assembly representation. Relationships between parts in an assembly are signified by what are referred to as ports. These are snake-like connections that criss-cross the assembly representation with information on pair-wise relationships between parts, such as physical connectivity.

Stahovich and Kara (2001) look at function and behaviour of components in design from a different angle. Their use of 'causal-processes' allows analysis of existing designs in an attempt to disseminate meaning from the presence of particular parts and then subsequently assign explanations for the existence of these components in a design.

3.4.3. Bond graphs

Bond graphs, developed by Paynter[13] (1961), Thoma (1975) and Karnopp et al. (2000), are used for synthesis by Sharpe (1978), Finger and Rinderle (1989), Broenink (1999) and Bracewell (2002). Bond graphs present a unified and flexible representation of physical systems by describing the transferral of energy through behavioural entities of a design in terms of effort and flow. Hence, as with input/output systems, bond graph representation is limited to systems with some aspect of flow. However, in such cases they are a useful tool for modelling the physical behaviour of complex systems, regardless of domain.

Bond graphs model engineering systems as energy interchange. Elements of a system are modelled as ports that are connected by power bonds. Ports are classed differently depending on their function: 1-ports dissipate or store power, e.g. resistors or capacitors, and 2-ports transform power, e.g. gear pairs. Analogous to junctions in electrical circuits, junctions in the graph (designated 0- and 1-junctions) are governed by generalised Kirchoff's laws (Gustafson and Wilcox 1998). Hence in a 0-Junction,

[13] Early in his career, Paynter was influenced by the work of Steinmetz (1909) and concluded that 'energy and power alone are the fundamental dynamical variables, the ultimate currency of all physical interaction and transaction'. This belief led to the creation of his bond graph representation, based on Couper's (1858) structural formulae for chemistry (Karnopp and Rosenberg 1968).

also referred to as a parallel junction, the effort in each bond is equal: the system has common effort. In a 1-Junction, also referred to as a series junction, the flows in each bond are equal: the system has common flow.

Ports, which can be labelled for computational purposes, are connected by bonds which represent an exchange of power. Power is the product of effort and flow, which act in opposing directions. Bonds are given a direction by a half-arrow attached to one end of the bond. This direction depicts whether power is entering or leaving the port through this bond. A causal stroke is added to one end of each bond to determine effort and flow: effort is defined as acting towards the end of the bond that has the causal stroke; flow is defined as acting towards the end of the bond that does not have the causal stroke.

Using bond graphs allows systems to be modelled and, potentially, designed irrespective of domain, as the energy flow model does not limit the model to a particular representation. Modelica[14] (Tiller 2001), an object-oriented software language for the modelling of physical systems, is based on a bond graph representation.

Welch and Dixon (1991) use behaviour graphs, similar to bond graphs, as a basis for generating solutions for conceptual design of mechanical systems. Bracewell and Sharpe (1994; 1996) have utilised a bond graph methodology to develop a knowledge-based design package called Schemebuilder. Tay et al. (1998) exploit bond graphs to generate and analyse dynamic systems for design of, for example, air pumps and vibration isolation mechanisms. The process involves using genetic algorithms to alter bond graph representations of possible solutions.

3.5. Analysis and evaluation

Quantitatively and rapidly evaluating the quality of a mechanical design is a challenging task. Being able to do this is a prerequisite for a successful generative design tool based on the parametric synthesis framework. In some design domains,

[14] http://www.modelica.org/ (last accessed 11 November 2003)

cost-based methods of evaluation can be beneficial despite their simplicity. For example, when building a simple mechanical artefact the main restriction on a particular project may be the cost of materials. In this case, designs can be evaluated based on the quantity and quality of materials. This information could be gleaned from a design relatively easily and employed as a useful quantitative measure of the 'goodness' of this design.

Apart from simple objective functions based on geometric properties of materials in a design, it is possible to investigate the performance of a particular design with respect to manufacture, assembly and subsequent disassembly for recycling. Data from such analysis can be used to estimate the quality of designs quantitatively. For example, 'design for assembly' (DFA) is a popular method of calculating such data based on estimated assembly times for particular parts in a design (Boothroyd and Dewhurst 1989). Other methods are 'design for manufacturing' (DFM) that includes information on component and supporting production costs (Boothroyd et al. 1994; Ulrich and Eppinger 1995), as well as more generic 'design for X' methods where the 'X' can be selected from a myriad of different possibilities, e.g. ease of disassembly for electronic equipment (Campbell and Hasad 2003). Such methods have been incorporated into evaluation methods for grammar-based generation (Agarwal 1999).

However, mechanical design problems are usually not as simple as this if one wishes to take into account criteria such as performance, ergonomics, environmental impact and aesthetics. More seriously, what if a design is unsafe? Even trained human designers and risk assessors have been known to fail when analysing designs for safety (Petrovski 1994). Additionally, as noted in section 3.4.2, designs that are perceived to be 'good' mechanical designs often exhibit a high degree of function sharing. Evaluating this degree of functional integration in a design is not a trivial task.

Some heuristics have been used relatively successfully for evaluation purposes in generative design. In his framework for conceptual design, Wahl (2001) uses the so-called 'skeleton model' as a preliminary measure of 'goodness' of one degree-of-freedom kinematic chains. This simple model ignores element volumes and measures the length of the functional path of the main design modules. By minimising this

distance, the skeleton model can be used to make a judgement on the potential of the concept being analysed by assuming a simple inverse relationship between length of skeleton and quality of design.

Such heuristic methods, however, side-step the most important evaluation criteria of all, namely the actual behaviour of designs. One of the strengths of Shea's eifForm tool (see section 3.3.1.1) is the ability to use inline evaluation in the form of finite-element analysis to judge the quality of designs 'on the fly'. This underpins the concept of performance-based synthesis (Shea and Starling 2003).

A mechanical equivalent of such analysis work would be to generate models as part of the generation process that can be analysed using behavioural modelling tool such as Dymola[15]. Dymola employs Kron's method of 'tearing' (Kron 1963) to solve multidimensional systems without resorting to linear simplifications. As an object-oriented modelling system, Dymola is suited to object-based synthesis using the modelling language Modelica (see section 3.4.3). Other domain-specific modelling toolboxes exist, such as Boeing's Easy5 tool[16] and Romax Technology's RomaxDesigner[17]. These tools can be used to produce simulation data to analyse design performance. The disadvantage of this sort of approach is the time it can take to run evaluation simulations, making evaluation from simulation less suitable for search algorithms that require a relatively high number of objective function calculations to be able to return beneficial designs.

3.6. Search and optimisation

Having investigated the possibilities for evaluation in design synthesis algorithms, the notion of search must be discussed in more detail. Simple, well-structured problems (Simon 1973), such as finding the minimum of a one-dimensional, continuous and differentiable function, can often be found analytically, i.e. a series of calculations can be undertaken to work out a precise answer to the question that can subsequently be verified. Such methods of finding optimal solutions are also termed deterministic.

[15] http://www.dynasim.se/ (last accessed 30 December 2003)
[16] http://www.boeing.com/assocproducts/easy5/ (last accessed 28 October 2003)

Given the same starting point, deterministic methods will always find the same optimal solution to an optimising search problem. Examples of deterministic search methods are exhaustive search, as well as gradient (Papalambros and Wilde 2000), branch-and-bound (Winston 1993) and pattern search (Torczon 1997) methods.

Unfortunately, deterministic methods cannot be expected to solve many of the design tasks one might wish to solve in engineering design. For example, the tractability of a problem depends on the number of variables, the type of variables (discrete, continuous, mixed), the size and extent of the search space, the objective function and constraints on the solution (Raphael and Smith 2003). Deterministic methods may be represented as tree-traversal problems. If all possible solutions are considered as leaf nodes of a tree, then moving from node to node will, at some point, lead to the correct solution. Unfortunately, the number of paths, i.e. links between nodes, explodes exponentially, i.e. for a tree of depth d (length of path to leaf nodes) with a branching factor b (number of parallel paths to descendant nodes) the number of solutions is b^d. In such cases exhaustive methods are not suitable as they would take too long to generate a viable solution, i.e. the problem has exponential complexity.

Non-deterministic methods, otherwise known as stochastic methods, rely on random numbers and are not always guaranteed to produce the same answer to an under-constrained design problem. In many cases, especially in synthesis tasks, stochastic methods present the best approach for finding optimal or near-optimal solutions to an optimisation problem. Such solutions are sometimes termed optimally directed solutions (Cagan and Mitchell 1993)[18]. Non-deterministic methods often rely on 'rules-of-thumb' to make acceptance and rejection decisions with regard to new solutions and so are often referred to as 'heuristic' techniques (Definition 3-8).

[17] http://www.romaxtech.com/ (last accessed 2 December 2003)
[18] Optimally directed design was first defined by Cagan (1990) to refer to search for designs in the region around the design optimum. This was redefined by Shea (1997) to 'design optimisation that directs […] design generation towards the numeric range of a global optimum'.

> ***Definition**: A heuristic technique (or simply, a heuristic) is a method which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee optimality, and possibly not feasibility. Unfortunately, it may not even be possible to state how close to optimality a particular heuristic solution is.*

Definition 3-8: Definition of a heuristic technique (Reeves 1996)

Examples of stochastic search methods are simple generate-and-test methods, such as Monte Carlo search. Other methods include simulated annealing (Kirkpatrick et al. 1983), tabu search (Glover and Laguna 1997), genetic algorithms (Holland 1975) and genetic programming (Koza et al. 2003). More detail on artificial intelligence and engineering design optimisation are provided by Winston (1993) and Papalambros and Wilde (2000). Deb and Jain (2003) use evolutionary algorithms for the optimisation of multi-speed gearbox layouts.

3.6.1. Constraints

Using constraints wisely is a very important part of engineering design. For example, it has been established that, all too often, an excessive restriction or too great an expansion of the search space is allowed during conceptual design (Fricke 1996). This can also be looked at from another angle – it is important to ensure that optimisation problems are 'well posed' (Papalambros and Wilde 2000). Ideally, constraints must narrow down the search space of a design problem enough to avoid endless, fruitless problem-solving. However, they must also not be too restrictive, as in so doing they may well rule out potentially useful design solutions.

For example, in some structural design cases it is often assumed that some sort of symmetry is advantageous. Hence, optimisation problems may often be set up to assume some level of symmetry as this can also make the optimisation problem less computationally intensive and hence more manageable. Shea and Cagan (1998) use a roof truss generation grammar to seek out asymmetric designs that are optimally directed and often better than solutions found when enforcing symmetry. In this case

setting a symmetry design requirement could have precluded finding good solutions by restricting the search space excessively.

Shea and Cagan (1998) use soft, as opposed to hard, constraints. Optimisation with soft constraints allows designs that violate some design constraints to be temporarily explored during the search process. This has the advantage that design solutions that lie close to constraint boundaries are easier to find using iterative optimisation methods. In cases where the design space is disjoint and non-convex due to the presence of forbidden regions, 'tunnelling' through these regions becomes possible to enable solutions across the complete design space to be found. The disadvantage of using soft constraints is that too much time can be spent considering designs that are outside the constraint boundaries. Even with increased tightening of soft constraints, it can sometimes not be possible to be certain of finding viable solutions. Using hard constraints can be more restrictive on the search process, but, by ensuring the consideration of designs within the constraint boundaries, can guarantee viable solutions, assuming any are found.

Constraints therefore ensure the validity of designs generated by the synthesis process. Bracewell and Johnson (1999) demonstrate the direct solution of a large set of design variables that are connected by various types of constraint. This work presents an alternative solution method for problems based on a component-based representation method that uses parameters and constraints to satisfy local optimisation problems (Thornton 1993). Interacting points of interest between components introduces connectivity and optimisation takes place through constraint satisfaction.

Similar work has been done by Schmidt et al. (1999) who are making an assembler to build on data produced by GGREADA as described in the previous section. The assembler uses a Constraint Satisfaction Problem (CSP) approach to produce geometrically legitimate designs from the functional and connectivity data provided by GGREADA. The vision of the work is to 'bridge the generation gap' (Schmidt et al. 1999) between abstract functional models and the use of synthesis methods to create legal configurations that satisfy function and form requirements for mechanical engineering design. Some success has been achieved. However, many of the designs

produced, while fulfilling the relevant constraint networks, do not fulfil mundane design objectives such as compactness, practicality and usability. An extension of this work utilises behaviour constraints for generative configuration design to help ensure suitable design variety in the generated solutions (Shi 2003).

3.6.2. Spatial packing problems

Spatial packing problems are an example of constrained design synthesis tasks based on what is known as the knapsack problem (Martello and Toth 1990). The idea is to pack objects into a confined space, i.e. the knapsack, using algorithmic means. A harder problem is the geometrically constrained knapsack problem (Cagan 1994), where objects in the knapsack are required to orient themselves in particular ways.

A variation on this problem specification is the class of routing problems where objects must be laid out and connected using cables or pipes. Szykman and Cagan (1995) have investigated such component layout problems. Cylinders and blocks are packed into a constrained volume using rules that allow object translation, rotation and swaps. This work was expanded to add pipe routing to the problem for application to HVAC (Heating, Ventilation and Air-Conditioning) systems (Cagan et al. 1996). Using a simulated annealing search algorithm (Kirkpatrick et al. 1983), dense packing was achieved for a heat pump example. Non-orthogonal pipe routing problems were solved for one-dimensional pipes, i.e. pipes without thickness. This work also investigated the packing of elements with spatial placement constraints for a battery-operated drill case study (Szykman and Cagan 1997).

Yin and Cagan have used a pattern search method (Torczon 1997) and grammar-based approach to solve complex shape packing problems, packing cogwheels with no connectivity constraints into a restricted area, as well as car engine parts into an engine compartment (Yin and Cagan 2000b). An extended pattern search method using a number of different heuristics to identify search directions provided faster convergence over previously used simulated annealing-based search for the same tasks.

3.7. Comparison of existing research

It is helpful at this juncture to outline a comparative snapshot of the main design synthesis approaches introduced in the previous sections. With a view to developing the storyboard design tool from chapter 1, four examples were chosen on account of the varied representations used and their impact on computational mechanical synthesis (Finger and Rinderle 2002; Li et al. 2001; Liu et al. 2003; Wahl et al. 2003). Sketches of the representations used can be found in Figure 3-5 and Figure 3-6. Note that the large arrow indicating levels of abstraction is meant as a guide and should not be used as an absolute measure for detailed comparison between the four examples.

Li et al. (2001) use a formal graph grammar to generate languages of epicyclic gear train designs. The new ring-plate-type cycloid drive in Figure 3-5 is a physical prototype that could not be represented by the original grammar. Following modification of the grammar, further cycloid drive designs were generated.

Liu, Bligh and Chakrabarti (2003) generate serial design configurations using a top-down exhaustive search with 'pruning' to limit results to potentially useful designs. Computational generation of physical solutions from spatial configurations is currently under development. Implementation issues are likely to include collision detection and scaling issues for anything more complicated than one-input, one-output systems.

The method proposed by Finger and Rinderle (2002) is the only one of the four to use a specific bond graph representation. Transformation rules are used to generate topological configurations from an initial specification. Synthesis is carried out explicitly 'by hand' and as yet this work has not been implemented computationally.

Wahl et al. (2003) evaluate one degree of freedom mechanisms using a skeleton model built from an extensive library of elementary blocks. A palette of possible design configurations is generated for a designer to consider. Optimisation is carried out at a high level of abstraction by minimising the length of the skeleton model and at a low level of abstraction through least-squares matching of the output motion compared to specification.

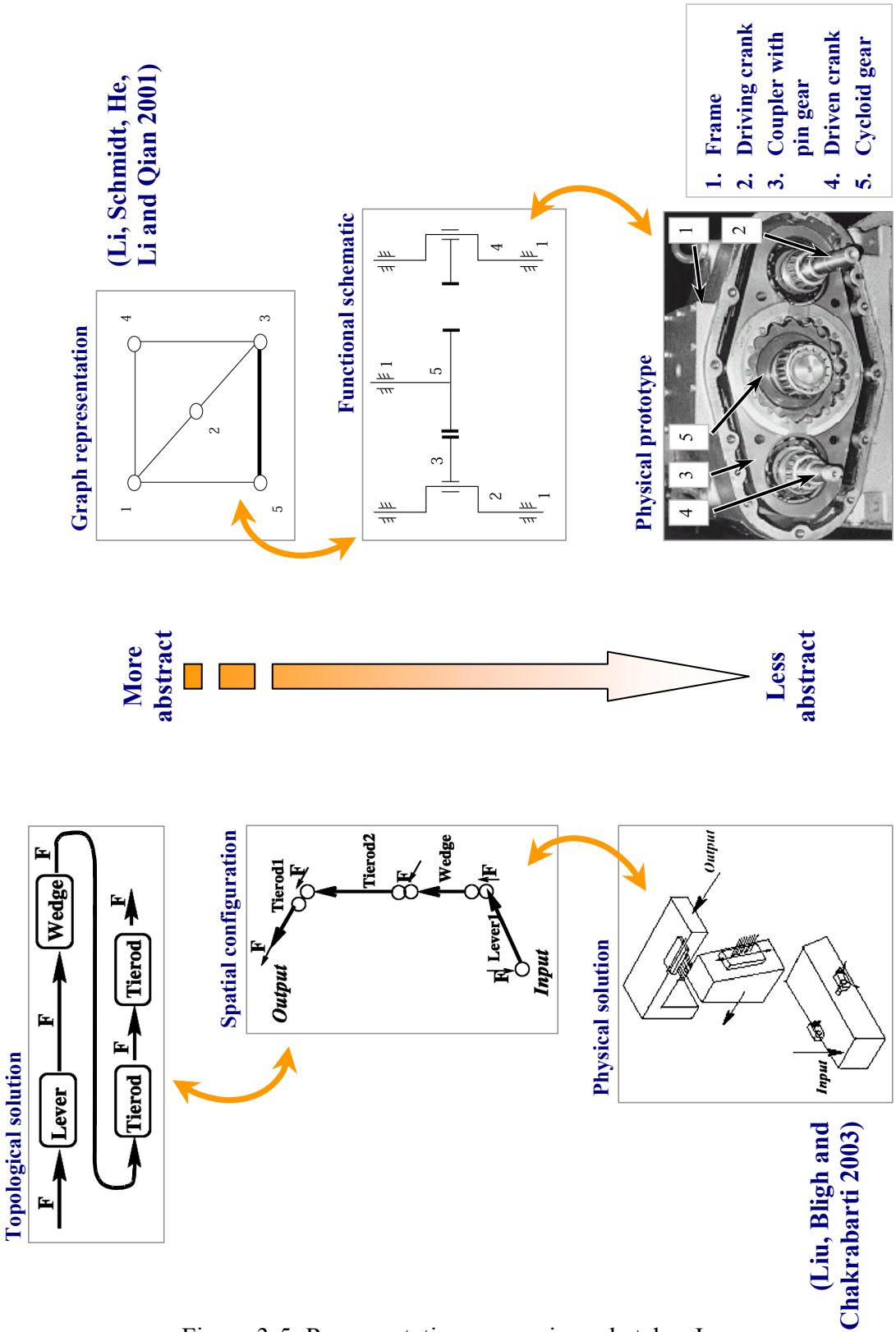

Figure 3-5: Representation comparison sketches I

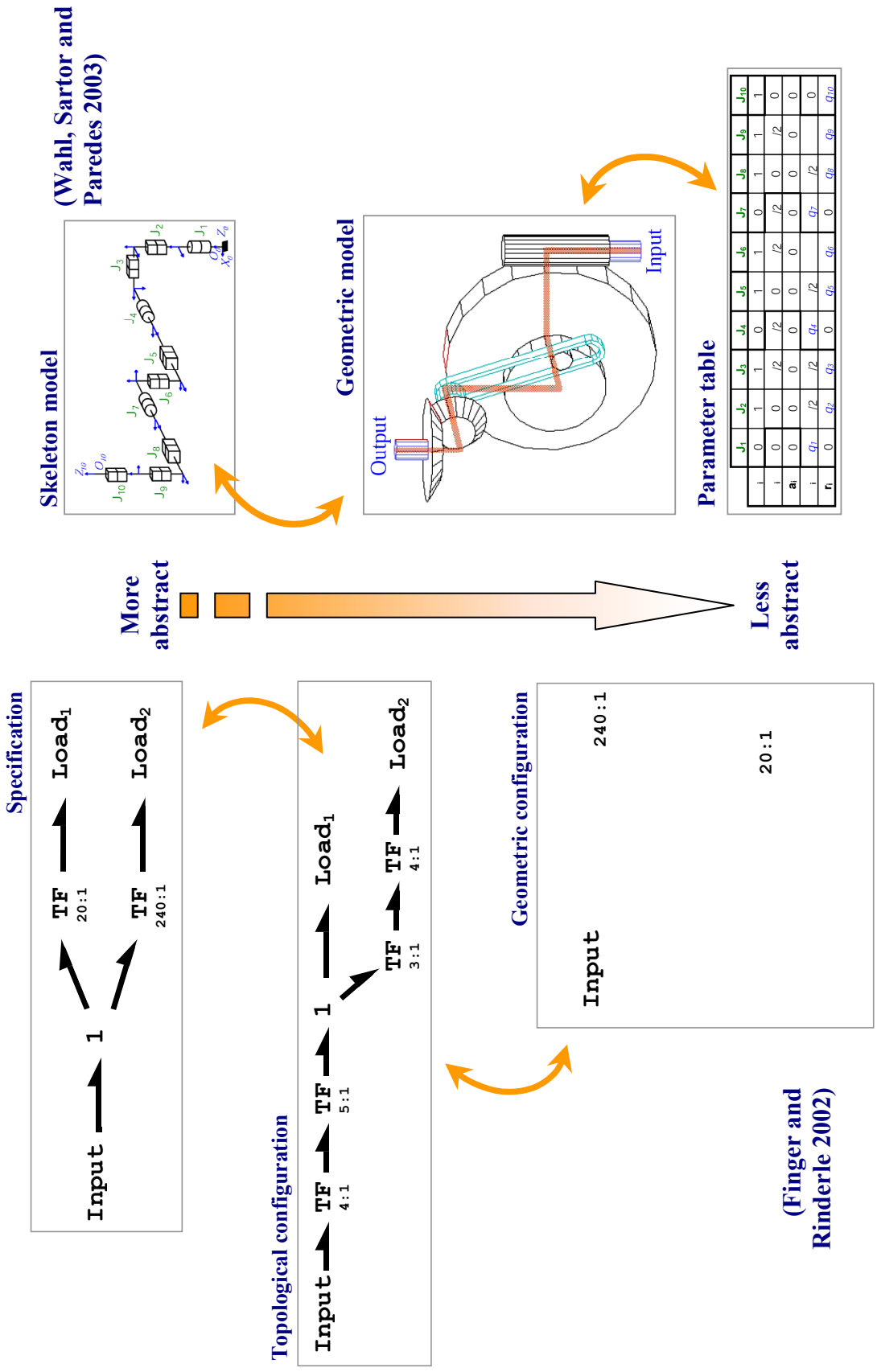

Figure 3-6: Representation comparison sketches II

3.8. *Thesis contributions*

A selection of background research has been summarised on a simplified 'design-line' (Figure 3-7). This attempts to plot the scope of research contributions with respect to the main Pahl and Beitz design phases (shown at the top of the diagram). Given the one-dimensional and simple nature of this sketch, it should be borne in mind that this graphic is indicative only. Some entries are broader than others and this does not necessarily mean that they subsume the content of smaller entries, rather that the research had broader rather than deeper implications on the design process. However, this does not detract from the intention of this particular graphic. It serves a useful purpose in allowing, at a glance, particular contributions to be 'mind-mapped' in relation to other work. The bottom entry ('Starling 2004') proposes the scope of research presented in this thesis.

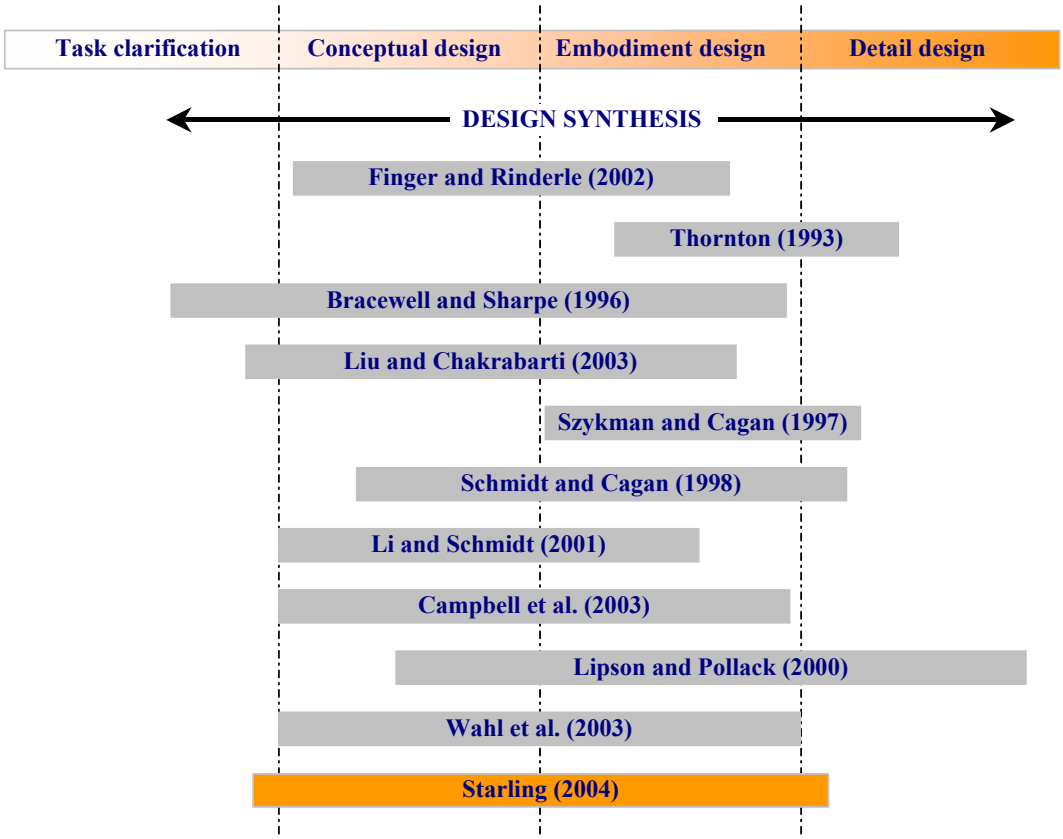

Figure 3-7: 'Design-line' comparison of existing work

This thesis concentrates on design synthesis of mechanical systems with a view to developing a synthesis formalism to underpin future computer tools as a possible basis for a mechanical design compiler (Ward 2001). This work is grounded in the parametric synthesis framework introduced in chapter 2. With respect to the literature reviewed in this chapter, the key areas of focus for this are as follows:

- Chapter 4: Design generation.

  A flexible and generic parallel grammar is developed to generate mechanical systems composed of gears, spindles and plates. The parallel grammar enables the synthesis of a language of mechanical designs from a library of parametric parts. Geometric and topological constraints ensure that only purposeful designs are contained in this language. Automatic generation of virtual prototype models is incorporated into the implementation of the parallel grammar.

  Parametric synthesis phases: *investigate* and *generate*.

  Contribution: a new parametric parallel grammar formalism is developed for computational form and function synthesis of mechanical systems.

- Chapter 5: Finding preferred solutions.

  The parallel grammar is expanded to include modification rules that facilitate the use of search algorithms to direct generation towards preferred designs. Design performance is quantified through geometry-based metrics to enable evaluation of designs to drive basic search for preferred designs.

  Parametric synthesis phases: *generate*, *evaluate* and *mediate*.

  Contribution: modification rules, an extension of the parallel grammar, allow basic search for preferred designs driven by geometry-based performance measures.

- Chapter 6: Enhancing design evaluation.

  Performance-based synthesis of mechanical systems is investigated by evaluation of behavioural simulation models that are automatically generated 'on the fly' as designs are generated. This introduces behavioural evaluation to the suite of performance metrics that can be used for seeking out optimally directed designs. The parallel grammar is used with a hybrid pattern search algorithm to generate a multi-objective palette of solutions for a camera winding mechanism redesign case study.

  Parametric synthesis phases: *generate*, *evaluate* and *mediate*.

  Contribution: performance evaluation is enhanced through behavioural simulation to close the loop on computational performance-based mechanical synthesis by enabling the automatic generation and feedback of analysis data.

- Chapter 7: Industrial applicability.

  Validation of the parallel grammar for parametric synthesis is provided by applying the method to power drill and vehicle gearbox design. The parallel grammar is enhanced to model clutches and allows more innovative exploration of potentially novel designs.

  Parametric synthesis phases: *mediate* and *investigate*.

  Contribution: the parallel grammar is enhanced to allow function graph modification, so providing a means of generating novel design configurations.

4. Design generation[19]

This chapter introduces a parallel grammar for the creation of mechanical systems. Based on a Function-Behaviour-Structure representation, the parallel grammar consists of two inter-related grammars for the generation of form and functional attributes of designs. The generation of clocks and watches is investigated as an implementation example of the parallel grammar. An existing clock design is recreated, using the parallel grammar, to verify the method. Novel clock layouts are then generated computationally, using a basic generate-and-test search method, to explore the design space described by the grammar.

4.1. A parallel grammar

It is desired to generate parametric mechanical designs to user-defined specifications using a computational tool. Synthesis approaches for such design tasks require a degree of abstraction to cope with domain-specific design issues, e.g. (Hoover and Rinderle 1989; Ward 2001).

In this work, a Function-Behaviour-Structure model (Umeda et al. 1990) is utilised in combination with a design grammar formalism (Chase and Liew 2001; Liew and Chase 2001). A parallel grammar is developed to create structures of mechanical designs that contain elements including, in the first instance, support plates, axles and gear disks. The parallel grammar consists of two grammars, (1) a function grammar that defines design connectivity and (2) a structure grammar that builds the physical parts of a design. Hence the function grammar is not dependent on spatial or form concerns. The 'glue' that links these two grammars is provided by geometric and

[19] An early version of this work is given in (Starling and Shea 2002).

topological constraints that ensure a consistent form and function representation for a design whose behaviour matches the requirements laid down by specification. These grammar constraints are discussed in more detail in section 4.4. A schematic of the parallel grammar is shown in Figure 4-1. The two grammars are referred to as being in 'parallel' as both are used simultaneously to generate a design: rule applications in the function grammar are followed by one or more rule applications in the structure grammar. Together, the two grammars produce designs that fulfil the constraints necessary for the required behaviour.

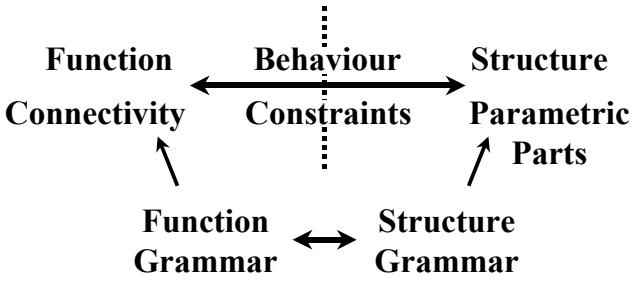

Figure 4-1: A parallel grammar schematic

4.2. The design domain of clocks and watches

Access to inexpensive, accurate and robust timekeeping is something that is taken for granted in the 21st century, but it was not always so. Throughout the millennia a variety of ingenious devices have been used for the purposes of keeping time, from simple hourglasses and sundials to more complex contraptions based on the flow of water. So-called 'clockwork'[20], mechanisms based on springs, ratchets and winding gears, was, and to some extent still is, a popular method of powering and controlling mechanical clocks. In fact, reliable clockwork mechanisms made for the first accurate positioning system for ships at sea[21].

[20] A clockwork power source stores energy in springs and imparts this energy to a gear train by effecting the rotation of an attached gear.

[21] In the days before inertial navigation and the Global Positioning System, determination of ships' latitude could be measured fairly precisely using a sextant, however, East-West measurements remained fraught with error. Parliament's Longitude Act of 1707 established a substantial financial reward for a 'practicable and useful' method of determining longitude. John Harrison, an English clockmaker, won this prize with a mechanical clock that could operate despite the heaving and rolling

These days more accurate methods[22] are available for keeping time, and the clockwork mechanism has been rendered obsolete as an indispensable technology. However, clock-making has continued to push technological advances. The demand for personal timekeeping devices, such as pocket-clocks and, subsequently, wristwatches, has led to many important innovations and advances in precision engineering, such as the use of jewel bearings for highly accurate positioning of mechanism components. The watchmakers' craft is highly skilled and is still a greatly respected art, to the extent that a large market still exists for luxury clockwork wristwatches that are worn as much for their functionality as for their image and the kudos associated with being able to afford such an expensive artefact. Certainly the most intricate examples of products from this industry boast a remarkable level of precise manufacturing. As an example, the IWC Shaffhausen 'Da Vinci' and 'Portuguese' wristwatches boast mechanical perpetual calendars with gear reduction ratios of 1 : 6,315,840,000. These watches will run with a high level of accuracy up until the year 2499 (Egginger 2003).

Today, consumer timekeeping devices take advantage of many different technological advances, such as electronic and mechatronic products that use quartz resonators to provide time lapse information. Different demands, skill sets and manufacturing options have resulted in a broad spectrum of products made to a variety of different specifications to fulfil combinations of secondary functions, e.g. being lightweight, accurate, in a particular cost bracket, of a particular size or having high perceived quality. In some cases, these secondary functions can become crucial to marketing strategy: the Swiss company Swatch emphasise the thinness of their wristwatches (Ashby and Johnson 2002), the current 'Skin'[23] range having a case thickness of a mere 3.9 mm.

Various clocks and watches, such as the clockwork alarm clock shown in Figure 4-2, have been analysed for this thesis using a reverse engineering approach (Otto and Wood 2001) to gain understanding of the design domain. A large quantity of non-

motion of ships and was accurate enough to be used to compare ship time with Greenwich Mean Time, so providing a method of determining longitude (Sobel 1998).

[22] In 1955 the first Caesium atomic clock was built at the National Physics Laboratory in the UK. Nowadays, atomic clocks have an accuracy way beyond that achievable by mechanical means.

technical data on clocks and watches is available in the public domain that, while of use, was not as valuable as information obtained from craftsmen, original patents from the US patent office[24] and published material from the International Watch Company[25], a Swiss corporation that produces clockwork watches for the luxury market. Clocks have also been used in academic research as examples for work in qualitative physics (Forbus et al. 1991).

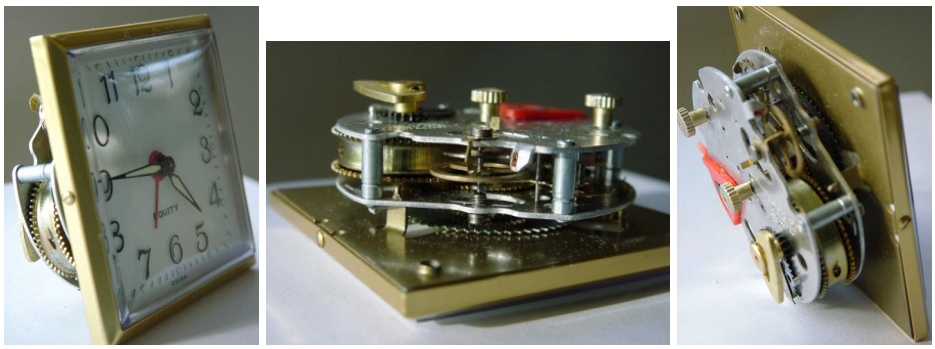

Figure 4-2: A clockwork alarm clock

From a technical point of view, the high-level function of a clock can be defined as follows:

> ***Definition**: The function of a clock is to keep track of the passage of time and display this information on demand in a fashion that is convenient to the user.*

Definition 4-1: The function of a clock

This definition can be used to establish system functionality using a 'black box' model (Otto and Wood 2001). It is common practice to establish three types of inputs and outputs: energy, material and information (Figure 4-3). The black box essentially represents the clock function (Definition 4-1) that can also be broken down into smaller sub-functions.

[23] http://www.swatch.ch (last accessed 13 January 2004)
[24] http://www.uspto.gov (last accessed 4 November 2003)
[25] http://www.iwc.ch/ (last accessed 4 November 2003)

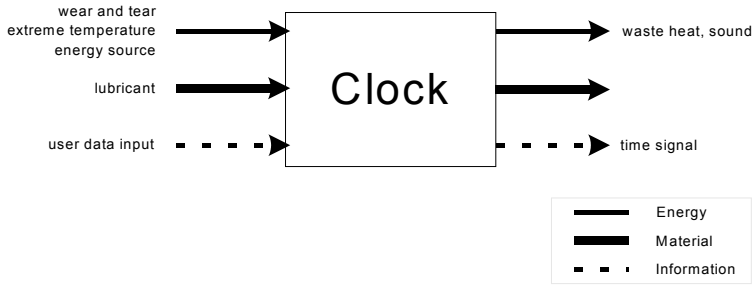

Figure 4-3: Black box model of a clock

Function-Behaviour-Structure (FBS) design models describe the relationships between function and structure, and link these to the behaviour of the design (Umeda et al. 1990). Information from the black box model in Figure 4-3 can be used to construct a simple FBS model of an analogue clock, shown in Figure 4-4 without product-specific details such as an alarm function. The product architecture is organised into a series of 'chunks' (Ulrich and Eppinger 1995).

A clear mapping exists between chunks and functions, for instance the clock face chunk has a direct link to the 'display time' function. However, inter-dependencies among chunks impact both behaviour and function so that these chunks cannot be designed in isolation. The desired behaviour of the clock is captured in the two requirements 'display time', the direct user-required behaviour, and 'have physical integrity'. This basic model was found to be valid for a large range of different types of clocks investigated: Table 4-1 shows a non-exhaustive list of comparative sub-function embodiments for a clock and a watch.

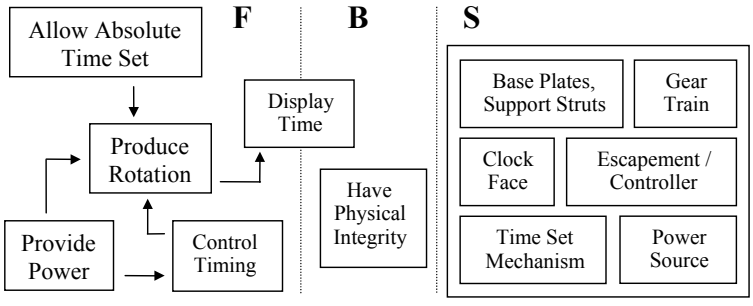

Figure 4-4: Simple Function-Behaviour-Structure model of a mechanical clock

Table 4-1: Sub-function embodiments for two different clocks

Sub-function	Electro-mechanical watch	Clockwork travel clock
Provide power	Lithium battery	Clockwork mechanism
Produce rotation	Gear mechanism	
Control timing	Piezo crystal used to provide timing for pulsed, quantised rotation	Mechanical escapement[26] in series with gear train
Display time	Clock face with hour, minute and second hands	Clock face with hour and minute hands
Allow absolute time set	Manual gear train control in parallel with standard power supply	

4.3. A clock grammar

The FBS model provides a good basis for developing the parallel grammar for design generation. Mechanical clocks are chosen as an initial demonstration problem for the parallel grammar as they have relatively straightforward functionality, as shown in Figure 4-4, but exhibit complex topologically and geometrically constrained parametric configurations, i.e. they are not just a linear gear train of standard parts.

4.3.1. The function grammar

The function grammar describes the connectivity of sub-functions in the design and is a grammar of the form:

$$G_{Function} = (V, X_F, R_F, S_F) \qquad \text{Equation 4-1}$$

$$X_F \in \{X_{F,N} \cup X_{F,T}\} \qquad \text{Equation 4-2}$$

$$X_{F,N} \in \{null\} \qquad \text{Equation 4-3}$$

$$X_{F,T} \in \{c, t, P, E, H, M, S\} \qquad \text{Equation 4-4}$$

$$S_F \in \{\spadesuit\} \qquad \text{Equation 4-5}$$

[26] An escapement is a type of ratchet mechanism that allows incremental advancement of a gear mechanism. When attached to a spindle oscillating in a simple harmonic manner, this device permits discrete advancements at regular time intervals, so providing a time control device.

V is a set of vertices with labels X_F in an directed acyclic graph (DAG) (Standish 1998). X_F is made up of a set of non-terminal labels $X_{F,N}$ and terminal labels $X_{F,T}$. R_F is the set of rules that are used to create and transform the function graph and S_F contains the initial symbol (see Table 4-2 for an explanation of labels used). The set of rules R_F are presented in Figure 4-5.

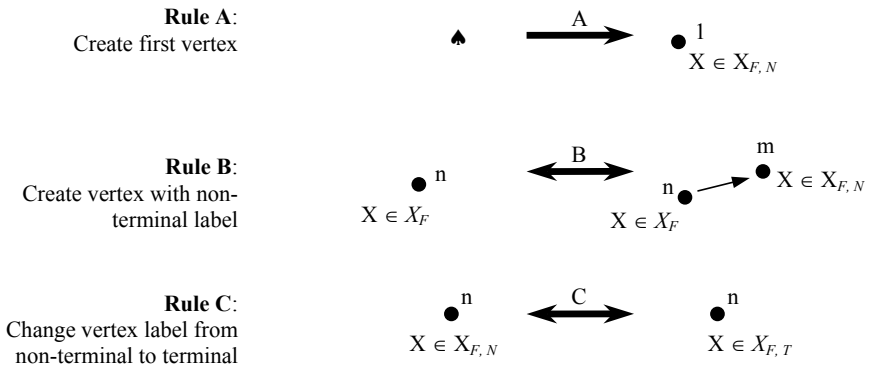

Figure 4-5: The set R_F of function rules

Rule A, the initial rule, is applied at the outset to create the first vertex in the graph. Rule B creates new vertices in the graph, adding the correct labels as required. Rule C allows the creation of terminal labels. Rules B and C are reversible to allow backtracking during the synthesis process. A function graph developed using these rules is shown in Figure 4-6.

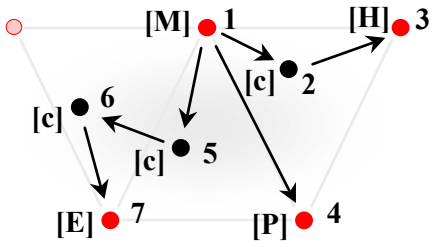

Figure 4-6: A possible function graph for a clock with minute and hour hands, escapement and power source. The connectivity labels [c] can, in general, be suppressed for completed graphs without information loss.

Table 4-2: Explanation of terminal labels used in the grammar

Label	Meaning
c	A vertex indicating further connectivity. A vertex with this label is a valid point to apply further structure rules.
t	A vertex indicating no further connectivity. A vertex with this label cannot have another connection added to it in the function graph through the use of rule B.
P	A vertex marking the power source in the function graph.
E	A vertex marking an escapement in the function graph.
M	A vertex that corresponds to the minute hand in the watch structure.
H	A vertex that corresponds to the hour hand in the watch structure.
S	A vertex that corresponds to the second hand in the watch structure.

For presentation purposes, the function graphs have been laid out in an ordered format to aid visualisation. Main terminal vertices are highlighted using colour and placed on a grid of isosceles triangles. Hence in Figure 4-6, the main vertices representing the hour and minute hands, as well as the escapement and power source, are represented by red nodes and laid out on a light grey grid. The faint node in the top left of the graphic is a placeholder for another potential vertex, for example as might be used to represent the sub-function of the second hand of a clock.

The directionality of the edges in the directed acyclic graph indicate the generation order of the elements of the graph. Vertices that are created before others will be connected by edges that point towards the more recent ones. It may be of more value to the designer to transform the graph so that the directed graph represents power flow through the design. In this case edges would represent effort in a manner comparable to the use of bond graphs (Paynter 1961). Performing this transformation on the function graph in Figure 4-6 results in one change: the direction of the edge between nodes 1 and 4 is reversed to indicate power flow (Figure 4-7). In general, this transformation can be implied by the interpretation of labels, i.e. [P] indicates power source and therefore power flows from this vertex.

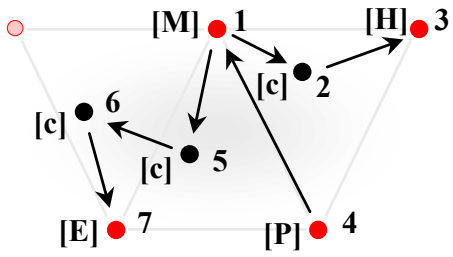

Figure 4-7: Transformed function graph to represent power flow

Each vertex of the function graph can be associated with separate axles, termed spindles in the context of clocks, in the gear mechanism. The function grammar determines the connectivity of these vertices, however, there is no explicit physical data associated with these nodes. This information is contained in the structure grammar, which is discussed in the next section.

4.3.2. The structure grammar

The structure grammar allows generation of a physical embodiment of a clock with connectivity determined by the function grammar. Structural elements include spindles, gears and support plates that are represented as parametric three-dimensional components. The structure grammar contains rules that generate these components, including their corresponding labels and spatial positions in the design, and are invoked in parallel with the function rules. Each particular function rule corresponds to one or more structure rules. Any parameters and spatial co-ordinates not explicitly determined by constraints and rule relations can be provided by the user or chosen randomly. The structure grammar is of the form:

$$G_{Structure} = (E_{Type,\,Data}, X_S, R_S, S_S) \qquad \text{Equation 4-6}$$

$$Type \in \{SPINDLE, PLATE, DISK, \ldots\} \qquad \text{Equation 4-7}$$

$$Data = \left[\begin{pmatrix} id_1 \\ \mathbf{r}_1 \\ \mathbf{R}_1 \\ r_{in,1} \\ r_{out,1} \\ \vdots \end{pmatrix}, \begin{pmatrix} id_2 \\ \mathbf{r}_2 \\ \mathbf{R}_2 \\ r_{in,2} \\ r_{out,2} \\ \vdots \end{pmatrix}, \ldots, \begin{pmatrix} id_N \\ \mathbf{r}_N \\ \mathbf{R}_N \\ r_{in,N} \\ r_{out,N} \\ \vdots \end{pmatrix} \right] \qquad \text{Equation 4-8}$$

$$X_S \in \{CON\_id, \ldots\} \qquad \text{Equation 4-9}$$

$$S_S \in \{\clubsuit\} \qquad \text{Equation 4-10}$$

The language of this grammar is defined by the structural elements $E_{Type,\,Data}$ where subscripts refer to the information that defines the structure. The subscript 'Type' carries information about the type of structural element, for example a gear disk is given the label $DISK$. The 'Data' subscript contains material properties, part features, local and global part co-ordinates, as well as identifiers for the N elements in the structure. Elements are assigned the same identifier id if they are part of the same physical object, hence a gear disk and the spindle it is attached to are assigned the same identifier since they correspond to the same vertex in the function graph. The labels X_S contain information such as which gear disks interact with each other. For example, a gear disk that interacts with another gear disk on spindle n would carry the label CON_n. As with the function grammar, S_S contains the initial symbol.

As the structure rules are used to create new components and add them to a design, structure rules are also referred to as 'Create Rules', or 'C-Rules'. The C-Rules, R_S, as implemented for the clock grammar, are described in Table 4-3 and presented in graphical form in Figure 4-8. This set contains a number of rules that are sufficient to generate the form of a basic clock from the starting symbol. Gear pairs are approximated by flat disks, or thick annuli if an axial hole is considered for the spindle, where the outer radius of the disk corresponds to the pitch radius of the gear. The reduction ratio of a gear pair is determined by the ratio of outer radii of the gear disks of the pair. This is a fair simplification if tooth size is small with respect to the dimensions of the gear. The structure representation in its current form therefore does

not consider such gear design details as addendum and dedendum values[27], nor does it consider the effects of tooth shape and contact ratio on the performance of the gear (Marghitu et al. 2001). Within the scope of this work, the refinement of the representation is covered during the detail design phase of the overall design process or is supplied by simulation tools (see chapter 6). The simplified representation is suitable for synthesis of concepts and general architecture.

Table 4-3: Description of structure rules

Rule	Description
1	Create a new spindle, replacing the initial symbol.
2	Add a spindle to the structure.
3	Connect two spindles with gear disks.
4	Add a base plate.
5	Add a power source (electronic) and connect two spindles.
6	Add a power source (clockwork) and connect two spindles.
7	Starting from two spindles, create an escapement.

Like the function rules, the structure rules, apart from the initial rule, are reversible. Hence during the course of a design, an object or collection of objects can be removed from the structure by finding it in the right hand side (RHS) of a structure rule and replacing it with the corresponding left hand side (LHS). Structure rules are invoked by function rules or can be executed manually. For example, adding a support plate to a design (rule 4) is not linked to a function rule. It is important to note that there is not a one-to-one mapping between function and structure rules, i.e. function rules invoke a non-exclusive subset of the structure rules.

[27] The addendum of a gear disk is the distance that the tooth profile extends outwards from the pitch radius of the gear. The dedendum of a gear disk is the distance that the tooth profile extends inwards from the pitch radius.

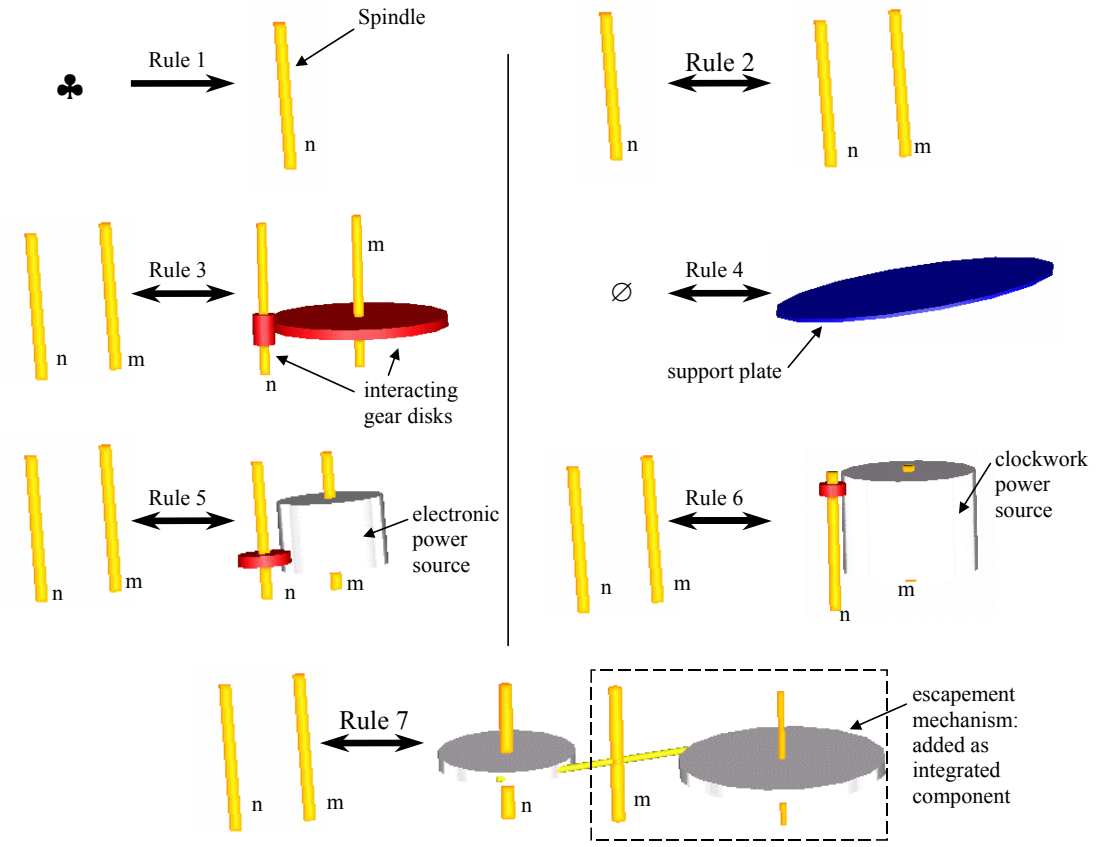

Figure 4-8: The set R_S of structure rules

4.4. *Constraint specification*

The importance of design constraints in synthesis was introduced in the background chapter. These are of special significance as the structure elements and rules outlined in this chapter are fully parametric. While this allows for variation in size of elements, it is also necessary for these parameter values to fulfil both general design and element-specific constraints, therefore ensuring the parametric and topological validity of designs produced by the grammar.

General design constraints ensure that no physical impossibilities occur in the structural design, such as two elements overlapping, i.e. sharing the same physical space. Element-specific constraints are more detailed, for example the 'mesh' constraint ensures that the two gear disks of a spur gear pair interact, i.e. the sum of the pitch radii is equal to the distance between the centres of the two disks. Another element-specific constraint, valid for all spindles that are attached to hands on the

clock face, is that these 'origin spindles' must be placed at the origin, i.e. have co-ordinates $X = Y = 0$, so that they can rotate around the same point to create a standard analogue display. This constraint is necessary to generate standard analogue clock faces. Relaxation of this constraint allows the generation of other possibilities, such as clock hands not centred on the origin.

In general, all constraints can be separated into two categories, topological and geometric constraints. All constraints are parametric to the extent that they control parameters in the design, however, topological constraints have a direct impact on connectivity and topology, i.e. they ensure functional requirements are maintained. For example the mesh constraint discussed above is a topological constraint in that gears are required to physically interact to uphold functional validity. A list of the main topological constraints can be found in Table 4-4; geometric constraints are listed in Table 4-5. In both tables a detailed description of the constraints is included. Figure 4-9 shows a gear pair schematic with related parameter annotations.

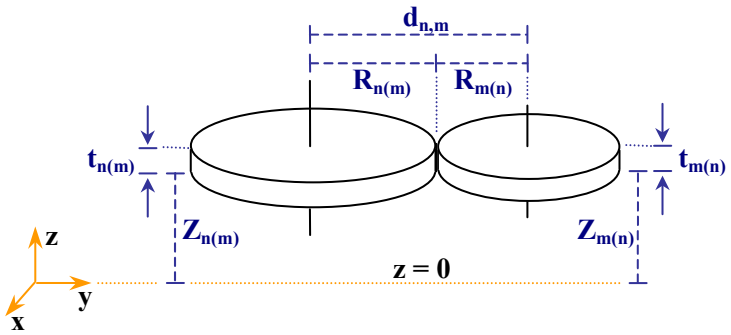

Figure 4-9: Gear pair schematic for constraint visualisation

Table 4-4: Topological constraints

Type	Detailed description
Mesh (plane)	Ensure that gear disks interact in the plane of the spur gear, i.e. that sum of outer gear disk radii is equal to distance between spindles that gears disks are attached to: $$d_{n,m} = R_{n(m)} + R_{m(n)} = \sqrt{([n].X - [m].X)^2 + ([n].Y - [m].Y)^2}$$ where *X*, *Y* are planar co-ordinates of spindles *[n]* and *[m]*, *[n(m)]* is gear disk on spindle *[n]* connecting to spindle *[m]* and vice versa.
Interact (axial)	Ensure that the parametric elements interact in axial direction. For example, for interacting gear disks: $$[((Z_{n(m)} + t_{n(m)}) \geq (Z_{m(n)} + t_{m(n)})) \; AND \; ((Z_{n(m)}) \leq (Z_{m(n)}))] \quad OR$$ $$[((Z_{n(m)} + t_{n(m)}) \leq (Z_{m(n)} + t_{m(n)})) \; AND \; ((Z_{n(m)}) \geq (Z_{m(n)}))]$$ where *n(m)* is the gear disk on spindle *n* that connects to spindle *m* and vice versa. *Z* is the absolute position of the gear disk from the *XY* plane, height of this gear disk is t_n. This constraint also affects interactions between gear disks and the spindles they are attached to.
Ratio	Ensure the overall ratio of mechanism remains unchanged. If an individual gear pair ratio changes, ensure that overall ratio of gear train remains unchanged. $$Let \; Ratio_n = \frac{R_{n(m)}}{R_{m(n)}}$$ $$Then \quad \prod_{n=0}^{N} Ratio_n = RATIO$$ where *n = 0, 1, 2, ..., N* are all gear joints that lie in series for which the total ratio must conform to a particular value *RATIO*. For example the gear joints between the hour and minute hands of a clock require a ratio of 60.

Free parameters, i.e. those that are not otherwise set by the constraints on the design, may be set by the user or selected randomly. As part of computational generation strategies that are introduced later in this chapter, the responsibility for setting these parameters is given to the algorithms controlling a search process.

Table 4-5: Geometric constraints

Type	Detailed description
Overlap	Ensure no overlap between parts within the parametric structure. A collision-detection library is used to check proposed changes and advise on geometric feasibility.
Boundary	Ensure that absolute and relative geometric parameters of elements fall within acceptable ranges, e.g. bounding box, minimum/maximum conditions: $Min\_Abs\_Param < [n].Param < Max\_Abs\_Param$ $Min\_Rl\_Param < [n].Param1 - [n].Param2 < Max\_Rl\_Param$ where *Param*, *Param1*, *Param2* are generic parameters, *Max_Abs_Param* and *Min_Abs_Param* are absolute spatial constraints, *Max_Rl_Param* and *Min_Rl_Param* are relative parameters.
Problem-specific	Equality constraints that are usually problem-specific, for example the constraint on the spindles that carry the clock hands requiring them to be concentric: $[n].Param = [m].Param$, where *Param* is a generic parameter.
Component	Minimum / maximum part numbers N, e.g. $N_{\min} < N < N_{\max}$

The hard constraints specified in Table 4-4 and Table 4-5 come into play during the application of grammar rules. Any addition or change to an existing collection of components may not conflict with any of the existing constraints. For example, adding a new component that would collide with an existing component is not permitted. This is determined in the implementation using collision detection software. As another example, consider the case of the ratio constraint. The grammar ensures that gear disks are sized suitably to create the desired ratio.

In chapter 5, the possibility of modifying existing designs is introduced, for example moving a spindle with an existing gear pair connection. Subsequent resizing of the gear pair – to uphold the ratio constraint – must result in a valid structure, i.e. an arrangement of components that does not violate any design constraints. Were this change to result in such a violation, e.g. a collision with another component, the grammar does not allow the change and the original, legal structure is restored. The use of constraints is generic and allows new gear sets to be added to the chain, the grammar ensuring that overall ratio-preserving gear radii are selected. The modification of existing structures is discussed in more detail in chapter 5.

4.5. *Verification of the parallel grammar*

The clock grammar was verified through the recreation of an existing clock configuration. This design was generated by hand-selecting function rules and choosing parametric values for the corresponding structure rules to reflect the original design. Figure 4-10 shows a generated model of the clock next to an image of the dissected clock shown in Figure 4-2.

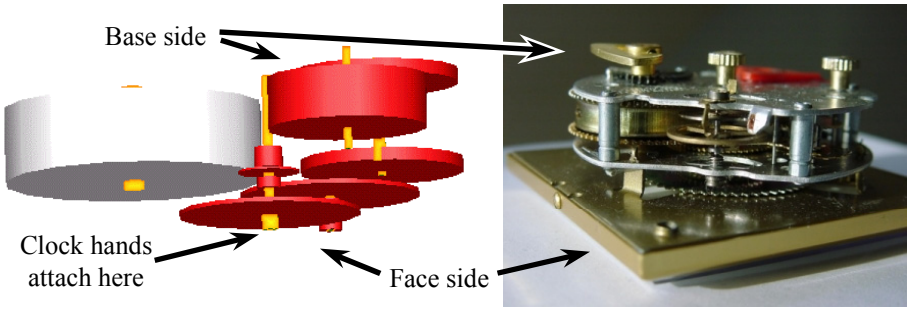

Figure 4-10: Synthesized model of real clock (base and face plates not included).

Figure 4-11 illustrates the generation procedure step-by-step, showing the transformations of the function graph and the physical structure by application of grammar rules to re-create the original design. Starting with their respective initial shapes, the first rules of both the function (rule A) and the structure (rule 1) grammars are applied to create the first vertex in the function graph and the corresponding first spindle in the structure representation. The starting spindle will be the minute hand and so the vertex in the function graph is given the label [M] by applying rule C (c.f. Table 4-2 and Figure 4-5). To aid visualisation, the same label has been added to the structure representation in Figure 4-11.

The structure is then built up further. Applying function rules B and C allows a new vertex to be added to the graph and labelled. This particular spindle will be an idler between the minute and the hour hands, so it is given the label [c] which has been suppressed in this graphic for clarity. In the structure grammar, rules 2 and 3 have been applied as part of this step to create the parallel structural components: A new

spindle has been added and then linked to the first spindle by a pair of intermeshing gear disks.

This procedure continues until the final design has been generated. Note how structure rules 6 and 7 have been used to create a power source and an escapement, the latter being represented by a standard spindle and gear disk. A second hand has not been included as the original clock (Figure 4-2) did not have one. Rule applications in Figure 4-11 are indicated by one-directional arrows to show the progression of the design. As all rules (apart from the initial rules) are reversible, they could equally have been represented by double-headed arrows to indicate that this is the case. Hence a series of rule applications can be undone, if required, to retrace synthesis steps back to an intermediate stage if the current design path is deemed to be fruitless or unsuccessful.

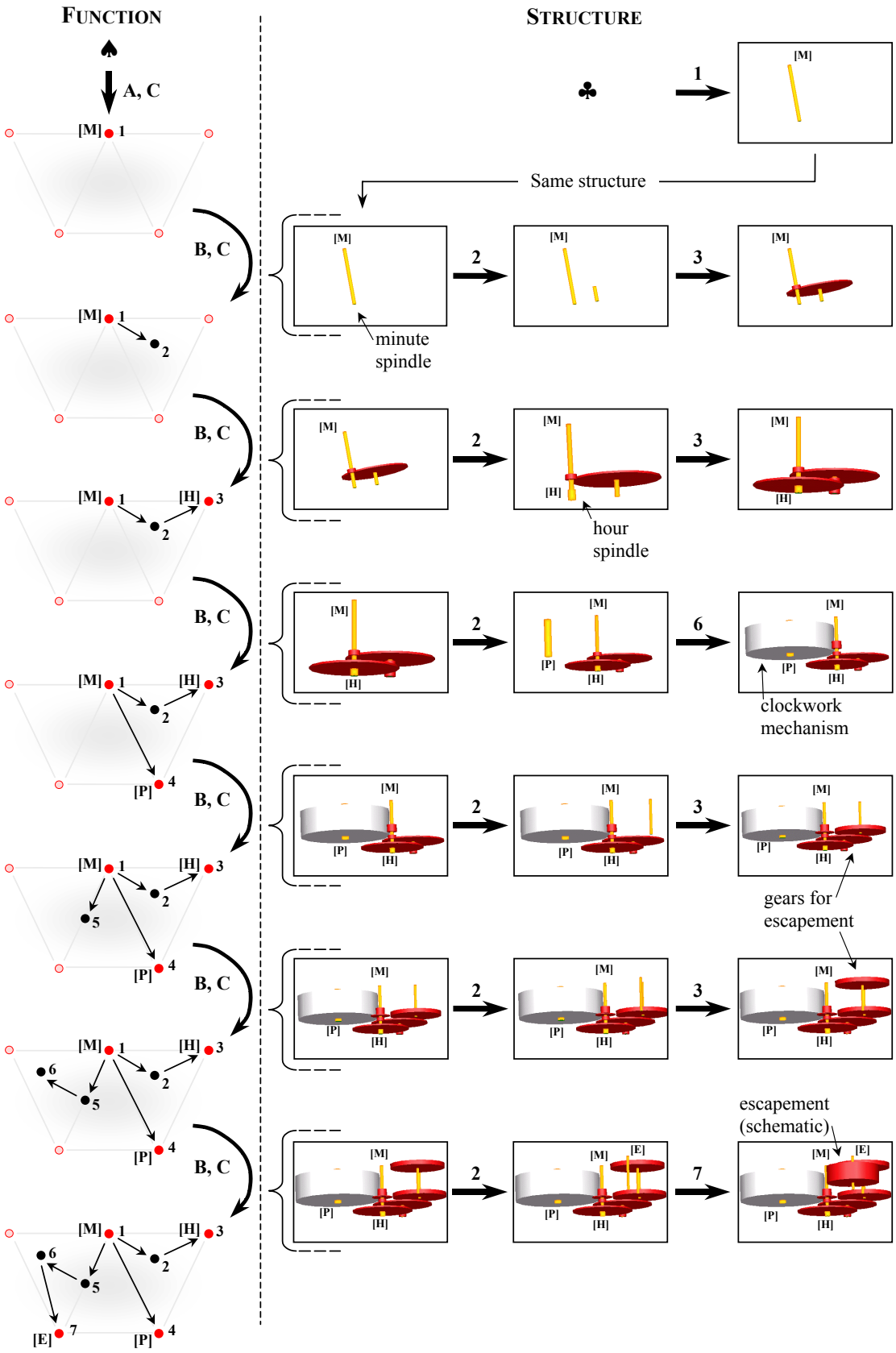

Figure 4-11: Application of rules to create a clock design. The colouring of elements in these diagrams has no functional purpose and is intended to allow spindle (orange), gear disk (red) and power source (white) elements to be easily distinguished.

4.6. Generation of clock designs

The successful implementation of this grammar demonstrates that a parametric grammar, that does not rely on pre-constructed fixed-size part libraries, can be used for the spatial and functional generation of a three-dimensional mechanical systems composed of gear disks and spindles. Further embodiments of this function graph can be created using a simple generate-and-test algorithm to generate designs that satisfy chosen design criteria. Solution generation is constrained by the design constraints, for example restricting the space available for the whole structure or limiting the maximum pitch radius of spur gears (Table 4-5). This section now explores some of the basic possibilities for using the parallel grammar for generation of valid designs.

4.6.1. Complexity and generality

The fact that application of structure rules can successfully create complicated designs from very simple descriptions is important for future development of this work. It is surmised that a grammatical approach to aiding complicated design problems cannot be efficient if very complicated and specific grammar rules are required as this would adversely affect the portability of grammar rules to other design tasks of a similar class.

As an example, consider, as discussed previously, the spindles that are attached to the hands of the clock. These spindles could be placed anywhere in the clock structure, however, to conform to the design specification that the hands of the clock must both lie on the same axis so that they rotate around the same point on the clock face, a clock-specific user constraint is imposed. This problem-specific constraint (Table 4-5) ensures that these so-called ‘origin’ spindles with labels that indicate hour, minute or second hands are placed at the centre of the clock cylinder and are attached to the support plate on the face side of the artefact to allow the attachment of the clock hands. This requires a more complex structure whereby shell-like spindles sit inside each other like Russian Babushka dolls.

Figure 4-12 shows how the same rules can be used to create different structures where the constraints condition the complexity of the structure. Three different final structures are shown, each of which was generated by the same sequence of structure rules but with different chosen parameters. From the starting symbol, rule 1 is applied to create a spindle, and rules 2 and 3 are applied to create two spindles connected by gear disks. The upper design shows the third spindle added as a shell around the initial spindle. The important point is that each of these three possible structural designs was created with the same structure rules, driven by constraints, to physically represent the same function graph.

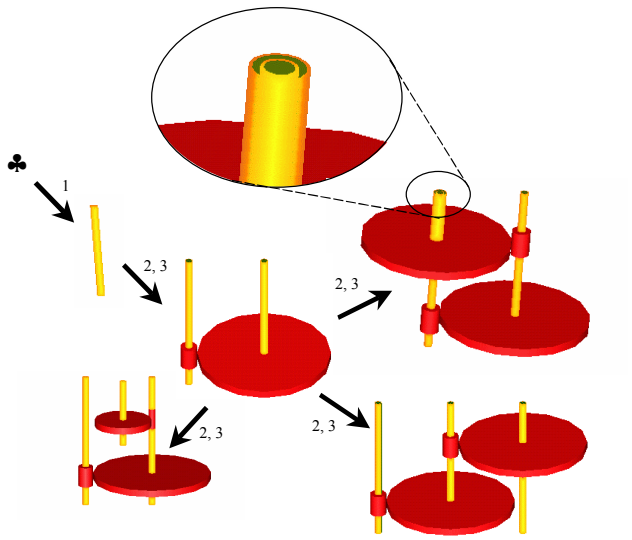

Figure 4-12: Application of structure rules

4.6.2. Generate-and-test

Having introduced the parallel grammar for the step-by-step manual production of an existing clock design (Figure 4-11) and investigated the low-level practicalities of using simple rules for building up different structures with the appropriate constraints (Figure 4-12), the possibility of computational generation of new designs was examined. The simplest form of computational implementation for such a grammar would be to prompt a user to provide a selection of function rules that, in turn, trigger the corresponding rules in the structure grammar. This data-entry process can be accelerated by allowing the user to input a set of function rules using an input file

with spindle and ratio data. A flowchart representation of the generate-and-test algorithm can be found in Figure 4-13.

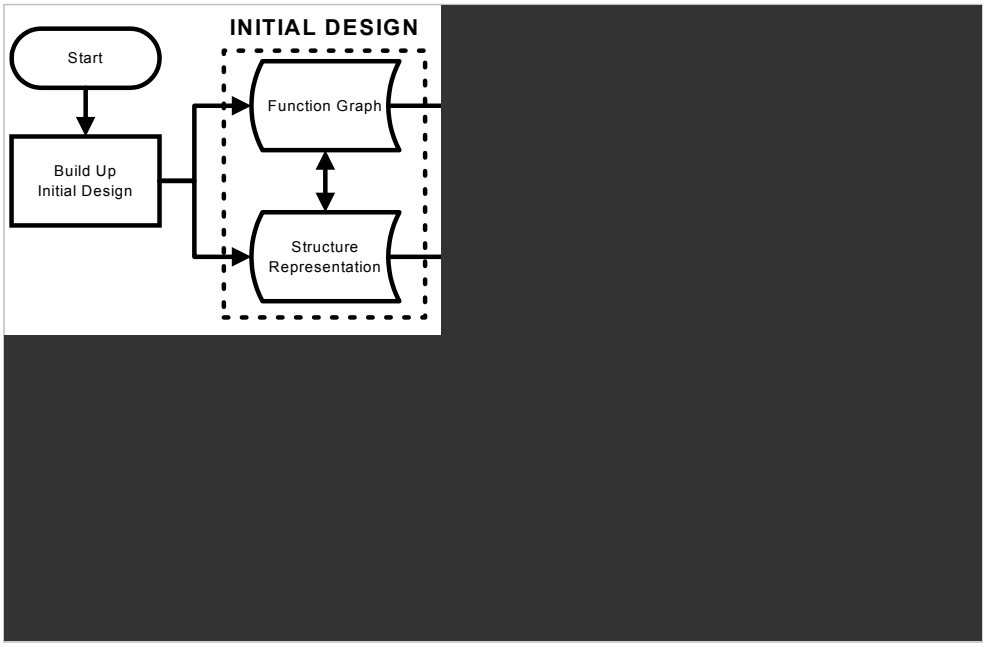

Figure 4-13: Generate-and-test algorithm for computational creation of design solutions

(shaded area contains synthesis steps that are added in the next chapters)

If these above-mentioned structure rules fire successfully, i.e. are able to create the relevant structures with the required parameters without breaking any constraints, a successful design ensues. On some occasions, however, structure rules will fail to complete successfully. This can occur in many situations, the simplest case being that the required parameters would break basic constraints, e.g. a newly added component may attempt to occupy the same space as an existing element in the design.

The generate-and-test algorithm uses random numbers to choose free parameters. Hence unsuccessful rule applications can occur relatively frequently, as the rules, which were designed to be as generic as possible so as to avoid highly tailored, complex grammars, include no knowledge of the likelihood of application success with regard to parameter ranges or application location within the design. In these cases, the 'brute-force' method employed by the generate-and-test algorithm repeatedly attempts to reapply a particular rule using a different set of randomly generated parameters until success is achieved or a pre-set maximum number of

attempts has been made. With respect to the original grammar, each repeated application corresponds to the grammar rules being applied in reverse to move the design back to a previous stage where all constraints are fulfilled, so restoring the validity of the partial design. Another rule application can then be attempted to try and move the design towards completion.

In cases where rule application is repeatedly unsuccessful, a 'return-to-base' strategy is employed to return to the starting symbol and restart the rule application process from scratch. Consider attempting to apply structure rules 2 and 3 to create a new spindle and gear pair of ratio 9:1 (see Figure 4-11 for examples of this rule combination). The spatial constraint on the smaller of the two gear disks will result in a minimum radius for this disk, with the 9:1 ratio resulting in the minimum distance between the existing and new spindles being at least ten times the minimum radius as specified by the spatial constraint (see Table 4-4 and Table 4-5 for constraints; see Figure 4-9 for gear pair visualisation). If this minimum size results in components that cause protrusion beyond the bounding volume of the design, then the results of this rule application will always fail due to the overlap constraint.

4.6.3. (Re)creating designs

The generate-and-test algorithm provides a straightforward method of exploring the language of clock designs described by the parallel grammar. Grammar rules were used in conjunction with the algorithm to generate further designs as a redesign task for the existing clock, shown in Figure 4-2. This design corresponds to the function graph in Figure 4-6.

Firstly, it was attempted to find a design for a similarly constrained space as the original clock. The volume for the clock was limited to a cylindrical space of height 20 mm and radius 30 mm, slightly bigger than the original dimensions. Additionally, each spindle was required to be anchored to at least two base plates with a relaxation on origin spindles to allow short 'stubby' spindles to be connected at just one end, as is found in successful existing clock designs. This design requires three connections to be made to the initial spindle [M]. This crowded configuration resulted in a low

success rate for the generate-and-test algorithm – 1068 attempts were required by the algorithm before a solution was found and this design is shown in Figure 4-14. The image is a picture of the virtual prototype generated as a VRML[28] model at run-time during the generation process. The 'stubby' hour spindle (#3 in the function graph) can be seen at the bottom, with the thinner minute spindle (#1 in the function graph) extending out above it. The gears for the power source and escapement can be seen nearer the top of the picture. General spatial constraints were set to conform to existing clock designs and are listed in Table 4-6.

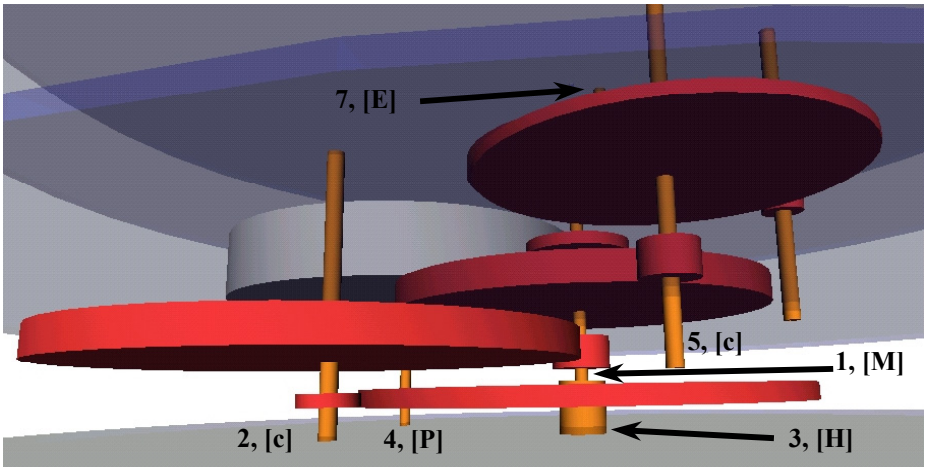

Figure 4-14: Solution set A – clock 1. Colourings are as before; in addition, plates are blue and semi-transparent. Labels have been added to assist in spindle identification. Spindle 7 ([E] indicates schematic escapement) is located behind spindles 5 and 6.

[28] Virtual Reality Modelling Language. VRML is an industry standard for virtual reality, solid modelling and general graphics work.

Table 4-6: Detailed spatial constraints for generate-and-test experiments

Parameter	Explanation and rationale	Value for sets A-D
R_{max}	R_{max} is an upper limit on the radius of gear disks to inhibit creation of infeasible part sizes (gear disk minimum radius provided by outer radius of spindles they are attached to).	40 mm
$RATIO_{min}$	These parameters provide upper and lower limits on $RATIO$, the ratio reduction of gear disks.	0.1
$RATIO_{max}$		20
$WALL_{min}$	Minimum radial thickness of thin-walled structures.	0.2 mm
$ANNULUS_{min}$	Minimum axial thickness of disk-shaped structures.	0.4 mm
$STAND\_OFF_{min}$	Minimum clearance required between non-interfering moving parts.	0.2 mm
$POWER\_T_{min}$	Minimum thickness of power source to provide enough space for mechanism.	4 mm
$POWER\_R_{min}$	Minimum radius of power source to provide enough space for mechanism.	10 mm

In a typical example of how the design process might take place, it was decided that requiring each spindle to be anchored at two points to base plates (coloured blue in Figure 4-14) might not be as important a criterion as reducing the overall thickness of the clock. Hence another experiment was carried out, requiring each spindle to be anchored at one point in the design, but reducing the thickness of the overall height of the design volume from 20mm to 10mm. All other constraining parameters were left unchanged. Despite the space constraints, the loosening of the anchoring constraint allowed for much faster design generation, with 20 possible designs generated in 82 attempts, of which the 3rd and 5th designs solutions generated are shown in Figure 4-15 (see Table 4-7 for comparison of constraint values for these different experiments).

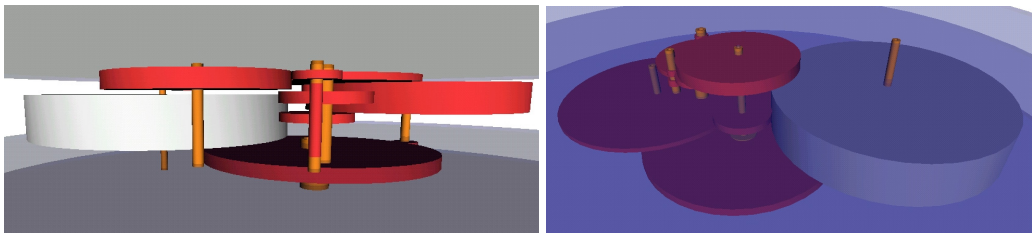

Figure 4-15: Solution set B – clock 3 (left) and clock 5 (right)

All the designs in solution set B feature a relatively high packing density of components to overcome the restrictive spatial constraints. Additionally, most parts are placed away from the clockwork power source (the white cylinder in the structure representations). The size of this component is restricted by the minimum values given in Table 4-6, as clockwork is quite bulky with respect to other parts in the design. The spatial restrictions are minimum values rather than equality constraints as the clockwork part is treated as a parametric component that can be sized according to the ratio between it and its connecting gear pair.

With the origin spindles restricted to be at the centre of the cylinder, the power source is one of the main spatial restrictions on these clock designs. It must not protrude beyond the confines of the base plates as this would violate the boundary constraint requiring the design to be contained within the cylindrical design envelope. Here the similarities between these two designs end. Solution 3 has arranged all but the gear pair between spindles 1 and 2 to be above the middle base plate, while solution 5 has a more even spread with the gear pairs between spindles 1 through 4 below the middle base plate.

Having generated new solutions for an existing design, a different clock design was considered that would not require an escapement, as is found in designs where the timing mechanism can be incorporated into the power source, for example in mechatronic clock designs. A second hand was specified for this new design and a possible function graph for the clock is shown in Figure 4-16. With relaxed height constraints on the clock (as might be found for a carriage clock type design that is placed on the mantelpiece of a living room) 20 new valid designs were generated after 72 attempts. Some of these designs, designated solution set C, are presented in Figure 4-17.

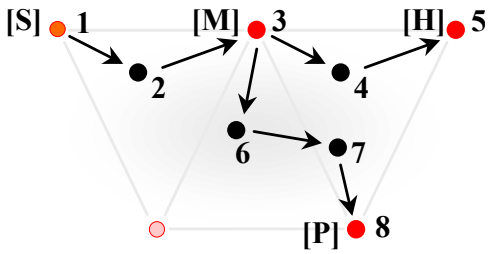

Figure 4-16: Solution set C – function graph

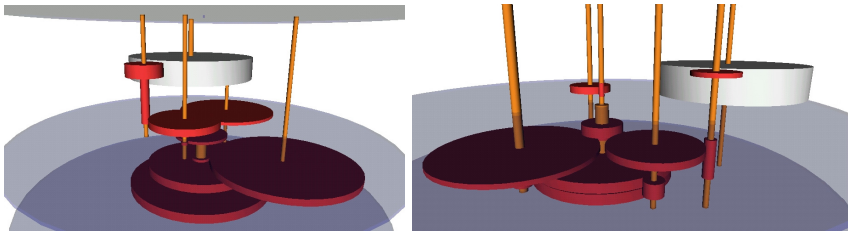

Figure 4-17: Solution set C – clock 5 (left) and clock 11 (right)

As a final experiment, designs were generated for a hypothetical function graph, shown in Figure 4-18. This function graph has a total of 13 nodes, a considerably larger number than those in the previous function graphs considered (7 for solution sets A and B, 8 for solution set C). Moreover, 5 of these vertices carry main terminal labels for the clock hands, escapement and power source. As in solution set C, the three origin spindles must all be concentric, and the power source is not directly attached to an origin spindle but connects to one of the idler gears between the second and the minute hand spindles. The rationale for such a design is that longer gear trains may be required if there is a low upper limit on the ratio of each gear pair, perhaps due to power transmission limits imposed by a particular type of power source or gear material. 20 valid designs were again generated for the more complicated function graph resulting in a success rate more akin to that for solution set A: 20 solutions from 2381 attempts. To illustrate the range of the language described by the grammar, four of these designs are shown in Figure 4-19.

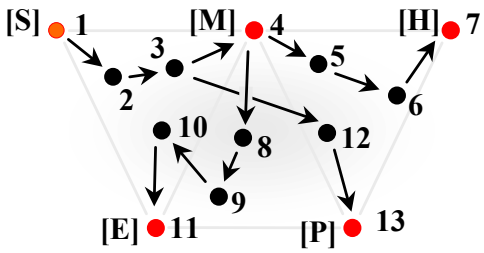

Figure 4-18: Solution set D – function graph (see Table 4-2 for explanation of labels)

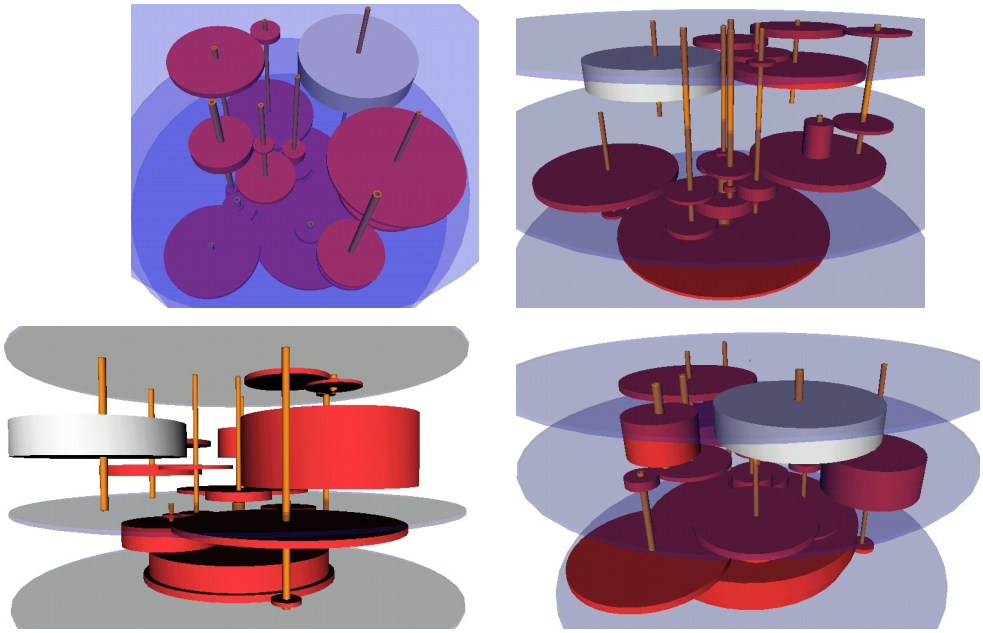

Figure 4-19: Solution set D – clock 12 (top left), clock 17 (top right), clock 18 (bottom left) and clock 19 (bottom right)

4.6.4. Comparison of generated designs

Visual inspection of virtual prototypes, i.e. VRML models, is important for design generation as it provides instant feedback on the quality and range of designs being produced. The information contained in such graphics, however, is only qualitative. A designer is not likely to want to wade through thousands of design prototypes to find the most likely candidates for developing a good design. The use of simple design metrics can be used to assign quantitative figures of merit to designs prepared by the generation process. An example of a simple metric is the mass of the design. With all parametric data available, it is a simple matter to calculate the mass of a design for given densities of material. Another simple metric is the aspect ratio of individual

components of the design, based on the principle of designing short, direct force paths (French 1999). Plotting these performance metrics for the generated designs can provide useful information to help direct the choice of suitable designs for investigation. Metric data for the design results discussed in this section was normalised to comparable values and plotted in Figure 4-20. The aspect ratio metric is based on a least-squares calculation that penalises extreme differences in aspect ratio (see Table 5-3). The depicted designs in the previous figures have been annotated.

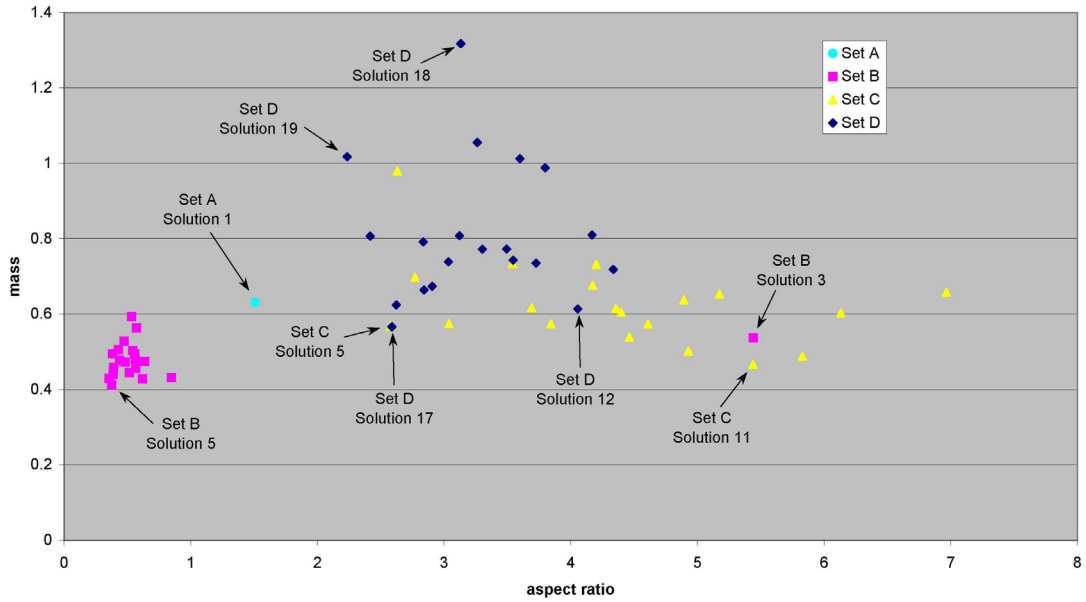

Figure 4-20: Simple metric representation of solutions for data sets A, B, C and D

The distinct grouping of the different sets of designs is a direct result of the different constraints set for the design scenarios. Set B has the strictest envelope of all the designs and hence only low volume (and hence mass) components can be selected during the generation process. This multi-objective presentation of the data also highlights particularly noteworthy solutions that merit further investigation, for example, solution 3 of set B has a particularly high aspect ratio metric value. Further analysis of the solid model generated for this design (Figure 4-15) reveals that this is consistent with a high number of flat, wide but relatively thin gear disks in this particular design. Design metrics are considered in greater detail in subsequent chapters. A list of the user-specified parameter values used for the generate-and-test experiment sets A-D are listed in Table 4-7.

Table 4-7: User-specified parameter values for generate-and-test experiments

Parameter	Detail	Set A	Set B	Set C	Set D
lower bound [mm]	Lowest point of any component in structure	-5	-5	-10	-10
upper bound [mm]	Highest point of any component in structure	15	5	20	30
clock radius [mm]	Radius of cylinder containing clock body	30	30	30	30
spindle attachment	Minimum number of plate-spindle anchored points	2	1	1	1

4.6.5. Clans and families of designs

Having generated several design examples for the grammar it is worthwhile investigating classification concepts with the design language. To what extent can two designs be considered 'different' or 'similar'? How could one describe this quantitatively? The design domain of clocks is investigated in this case, but the concepts introduced here will be applied to other domains in subsequent chapters.

The generated designs in sets A and B vary parametrically, but the underlying structure is similar as all designs generated have exactly the same functional representation (Figure 4-6). These designs can be considered to be members of a 'family' of designs. They have the same function graph and designs are distinguished by having different parameters.

On a more abstract level, there are clear differences between these designs and those in set C and D. Not only do the designs vary parametrically, but their fundamental connectivity as specified in the function graph makes them distinct. In a similar manner to class-based representation systems, these designs can therefore be thought of as being in different 'clans', a level of abstraction higher than a family of designs. This clan hierarchy is represented in Figure 4-21.

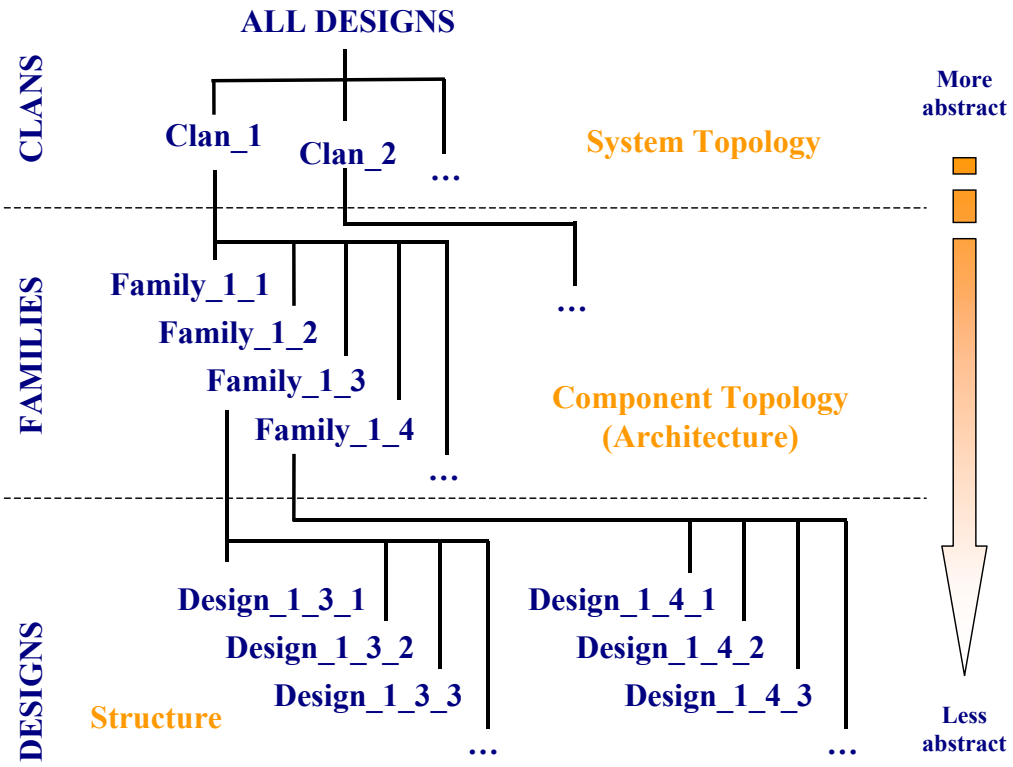

Figure 4-21: The clan hierarchy

Clans, at the top of Figure 4-21, differ in their system topology – main connectivity lines are all that need be considered to decide whether a design lies within a clan. Each clan contains many families, i.e. possible function graphs that can fulfil the general clan system topology. These families, the middle layer in the figure, differ in component topology. Each family, represented by a particular function graph, contains many possible designs, i.e. possible parametric variations on the design parameters specified in the structure representation of the function graph. Figure 4-22 shows a similar graphic to Figure 4-21 but with example designs taken from those explored in this chapter.

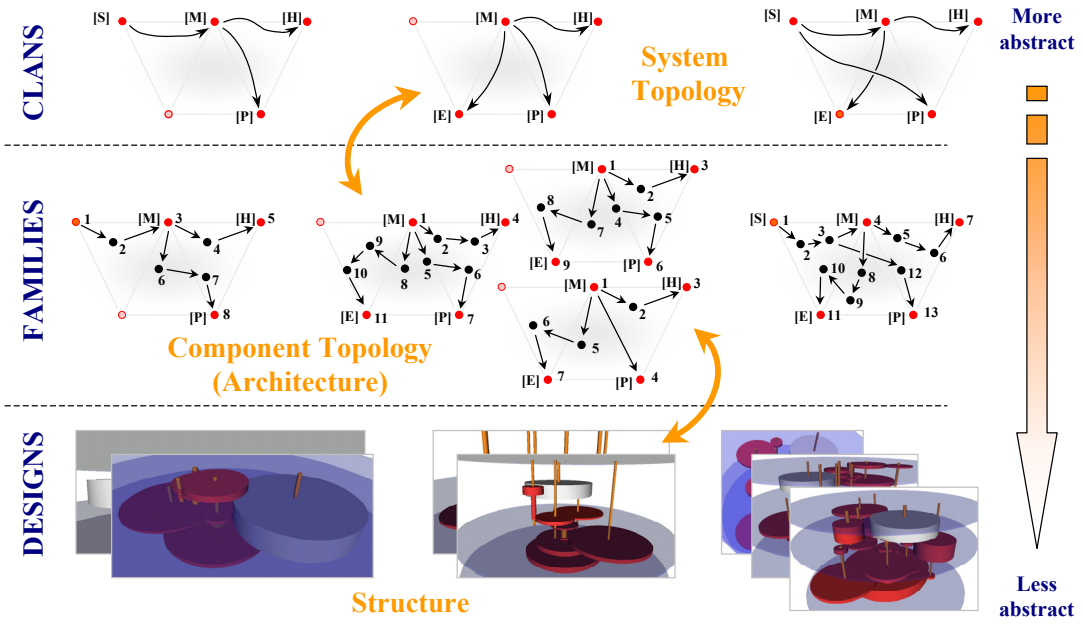

Figure 4-22: Clans and families of designs – some examples

4.7. *Conclusions*

This chapter has presented the creation and use of a parallel grammar based on an FBS design model for mechanical design synthesis. The grammar has been used to generate clock designs using three-dimensional parametric parts while adhering to a set of geometric constraints. The implementation of the parallel grammar has shown that the grammar-based design representation can produce valid designs and, in combination with a generate-and-test algorithm, can produce clock designs targeted at varying design criteria.

The parallel grammar presented aims to synthesise designs with required behaviour by simultaneous consideration of both the function and structure of a design. Topological synthesis of mechanical designs in the clock domain has been demonstrated using parametric parts to create geometrically constrained solutions to a simple problem, showing that constraints can be used to control the application of simple function and structure rules to produce different solutions.

4.8. Limitations

The ultimate aim of this research is to develop methods and tools that can be used to assist innovative mechanical systems design. However, many improvements are still required. The parallel grammar provides a sound basis for developing a usable design tool. The generate-and-test algorithm introduced above is relatively simple: the system relies too heavily on brute force and does not exhibit good scalability, e.g. increasing design complexity by a small degree (solution sets A and D over sets B and C) results in a much lower success rate. This could be addressed by increasing the amount of knowledge in the grammar rules, perhaps allowing grammar rules more autonomy to consider their immediate environment before they are applied at a particular point in the design. The stated aim, however, is to keep the grammar rules as simple as possible to ensure flexibility of design generation and avoid task-specific and narrowly defined grammars more akin to knowledge-based systems, so this route will not be explored.

The structure rules in the parallel grammar are effective for building up possible designs from scratch, but show a high degree of inflexibility once a design has been generated. For example, each of the designs plotted in Figure 4-20 was generated from scratch. Consider solution 3 of set B in this graphic that is far removed from the remaining 19 designs in this set. On being presented with this data, the designer might well want to investigate this particular design further and explore the design space immediately adjacent to it. Currently, the designer would have to do this manually as there is no way to simply modify parameters in the structure representation using the grammar, as all structure grammar rules require addition or removal of components in the design. Hence there is no simple method of lateral exploration of designs within families and therefore no way of applying a simple search algorithm to locate good, or even optimally directed designs, in this local design space.

The generate-and-test algorithm is able to generate different designs for a given family of designs. To generate a design of a different family, a new function graph must be specified by the user. This is rather inflexible. A good exploration of many different families and clans of designs would require the user to specify a large number of function graphs separately. It would be advantageous if some sort of

modification could take place that would allow designs to be perturbed to move between families. This would require a mechanism to remove and add nodes in a function graph without altering the system topology specified by the clan. It would also be useful to be able to generate different system topologies to generate even more possible designs.

In summary:

- A method for perturbation within design families is required to enable local exploration of designs within the language.

- The generation process for creating designs is wasteful and too naïve.

- A method for comparison of valid designs is required to improve the mediation between design alternatives.

- It would be interesting to allow perturbation of designs to move between families and clans to explore a greater search space than is currently considered.

The first three issues are addressed in chapter 5, the last point is taken up in chapter 7.

4.9. Implementation details

The parallel grammar has been implemented in a Linux-based C++ environment, where the function and structure grammar are instantiations of two main classes. The function graph is stored as a matrix that holds all edge, vertex and label information. The structure representation comprises a linked list of instantiations of a general element class. Labels and parametric data are stored within this format. Design solutions are saved in main memory while the program is running. Data and models are written to file upon completion of the generation procedure.

A collision detection library, FreeSOLID[29], is used to assist geometric constraint satisfaction. With this library, constraints pertaining to the interference of objects do not have to be described in the code explicitly, so making the process of constraint checking more transparent. Designs generated by the system are exported as Virtual Reality Modelling Language (VRML) files. These virtual prototypes are interpreted by VRML viewers as three-dimensional design models that can subsequently be visualised, inspected and analysed. A screenshot of a VRML viewer is depicted in Figure 4-23. Cursor-controlled manipulation allows rotation and translation of the viewpoint to enable visual analysis of the model.

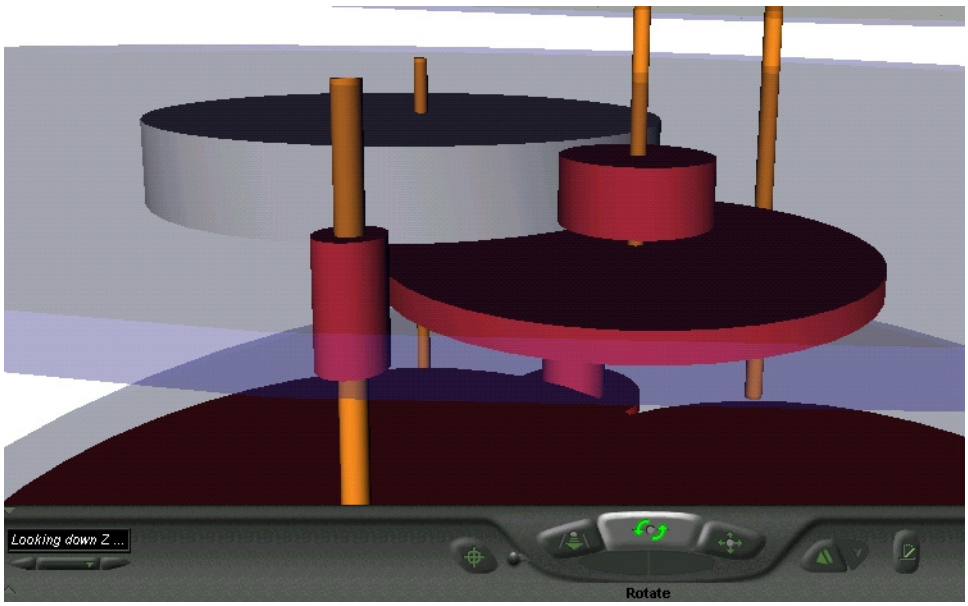

Figure 4-23: Inspecting a generated model with a VRML viewer

Time taken for solution generation varies with the complexity of the function graph. More straightforward solutions, for example sets B and C, are generated in time of the order of seconds, for example, the 20 solutions for set B were generated in under 4 s of user time, 1000 solutions for the same problem were generated in 143 s of user time[30]. Problems with more restrictive constraints and more complicated function graphs required a greater number of iterations, pushing the user time into the order of minutes for generation of 20 designs.

[29] http://www.win.tue.nl/~gino/solid/ (last accessed 7 November 2003)
[30] This data was generated on a PC with a CPU running at 1200 MHz.

5. Finding preferred solutions[31]

This section introduces a two-pronged strategy to address the main improvement issues raised in chapter 4 relevant to the generation of mechanical designs using the parallel grammar. The approach presented (1) enhances computational generation of new designs and (2) introduces the possibility of quantifying the quality of such designs.

A modification method is introduced that enables the perturbation of parameters in designs, therefore allowing the exploration of the design space local to a particular partial or complete design. This extended method uses a new set of structure grammar rules called perturb rules (P-Rules) that provide a means for enhancing the existing generate-and-test algorithm. In situations where progression to a complete design has faltered due, for example, to an ineffective layout of parts, the perturb rules allow the spatial rearrangement of such problematic layouts to enable application of further generative grammar rules and therefore the resumption of successful design creation.

The application of perturb rules does not affect the system topology of designs. Both the LHS and RHS design states of a perturb rule application can be compared in a like-for-like manner, introducing the possibility of using design metrics to mediate between designs to provide a basis for searching out preferred design configurations. This is the second main contribution of the work presented in this chapter.

[31] An early version of this work is given in (Starling and Shea 2003).

5.1. Modification of designs

The main achievement of the previous chapter was to demonstrate the successful generation of valid designs using a grammatical approach. The parallel grammar rules presented add and delete elements in their respective representations to move between initial and final design states. Application of this set of grammar rules can be imagined as large-scale changes to the design state, i.e. a part is always added or removed.

5.1.1. Rationale for perturbation grammar

As part of the general synthesis process, it would be useful to be able to make more minor changes to designs. Referring back to the family and clan classification introduced in chapter 4 (Figure 4-21), being able to create designs of one family by moving directly from design to design through parameter modification would be more convenient than having to remove existing components and replace them with parametrically different parts. Being able to change the parameters of existing elements in the structure representation, without actually adding or removing parts as above, would be a useful augmentation of the parallel grammar. Two main types of situations are observed where access to such an extended set of grammar rules may be of benefit.

The first requirement for design modification is as an aid to the generation of new designs. During the process of building up a design from scratch, application of structure rules to add new elements might fail due to an unfavourable component layout in the existing structure. In the clock problems in chapter 4, consider attempting to connect a new gear pair to the inner spindle of two concentric spindles. If the inner spindle does not protrude significantly from the outer spindle ‘sleeve’, this rule will be destined to failure from the outset if the parameters of the existing structure cannot be altered.

The second main need for a modification process stems from the desire to explore the local design space. For the generative design process to be of fundamental use to a designer, merely producing designs that satisfy parametric and topological constraints

is not enough. Having generated many possible solutions, it is also paramount to be able to search for preferred design solutions. This equates to being able to explore the local solution space of known valid designs.

5.1.2. Modification issues

Modifying designs equates to changing the parameters of design solutions. As an example, consider a simple modification rule that adjusts the global position of one end of a spindle or axle. This is a relatively simple modification to make, but the change may only be permitted if the new design state does not violate any constraints, such as making the object longer so that it collides with another object (collision constraint), or making it shorter so that gear disks on this spindle become detached (interact constraint). Hence changing one simple parameter of such a component can cause the design to become invalid.

Other design modifications are more complicated than this. Moving the location of a spindle in space requires gear disks on this spindle to be moved as well. The ratio of the gear pair is required to be unchanged, resulting in a resizing of the gear disks on the spindle, as well as those on connected axles. This knock-on effect means that an initially successful modification (i.e. the original axle movement in this example) can actually fail due to changes occurring beyond the scope of the initially altered component (i.e. the subsequent gear resizing operation).

5.1.3. Expanded generation framework

An extended generation framework is presented in graphical format in Figure 5-1. The core remains from Figure 4-13. Two major new decision boxes are added for design modification and evaluation. Initial designs are modified and evaluated as part of a cyclical synthesis process that results in improved designs.

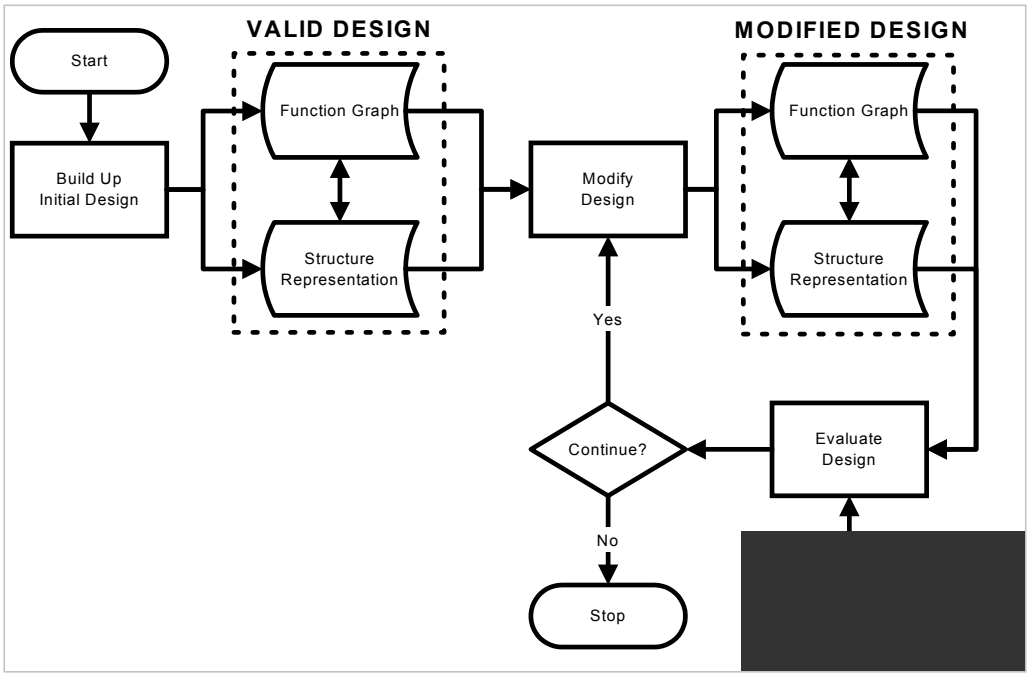

Figure 5-1: Finding preferred designs – modification and evaluation
(shaded area refers to updates introduced in chapter 6)

5.1.4. Perturb rules

The issues raised by parameter modification in designs have been discussed in section 5.1.2. Hence modification of design parameters is proposed through the use of perturb rules (P-Rules), a new set of grammar rules that complement the existing C-Rules of the structure component of the parallel grammar. These perturb rules allow the variation of existing parametric structures as they facilitate parameter changes while ensuring constraint satisfaction. A set of 14 perturb rules were created and these can be seen in Figure 5-2 and Figure 5-3.

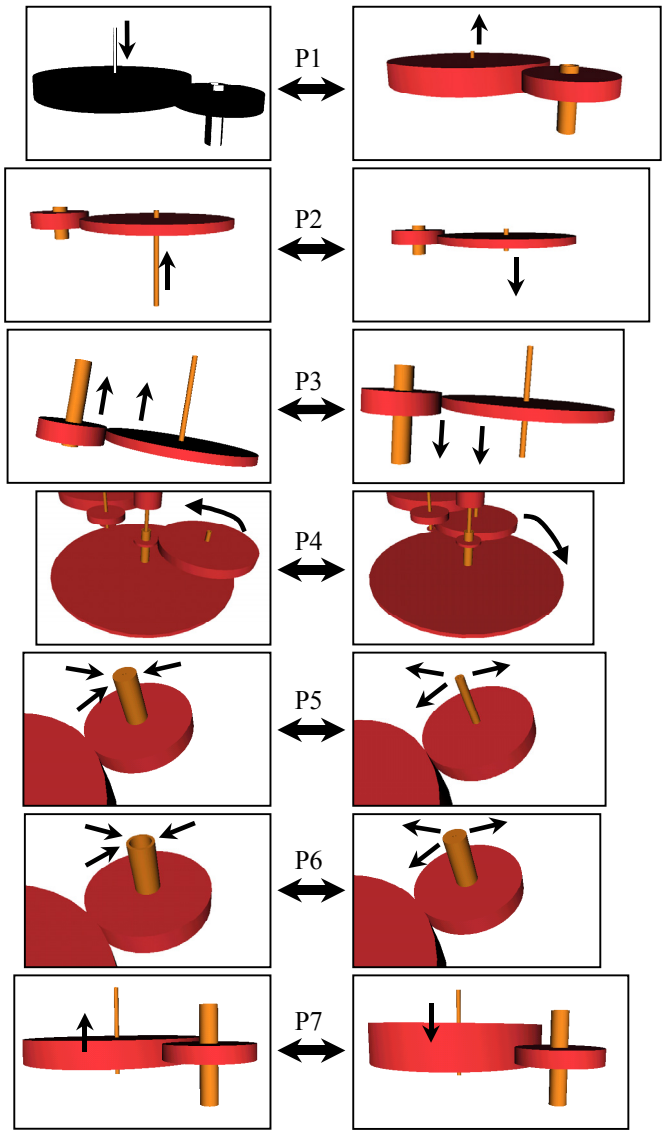

Figure 5-2: Perturb rules I

Table 5-1 and Table 5-2 give detailed descriptions of each of these rules including the rationale behind their use for modifying designs, including key constraints that impact their application. Compiling a portfolio of P-Rules is not meant to be an arbitrary exercise and therefore this rationale is an important part of understanding their particular impact on designs. This set of perturb rules should be thought of as a base library that can be extended for new design problems.

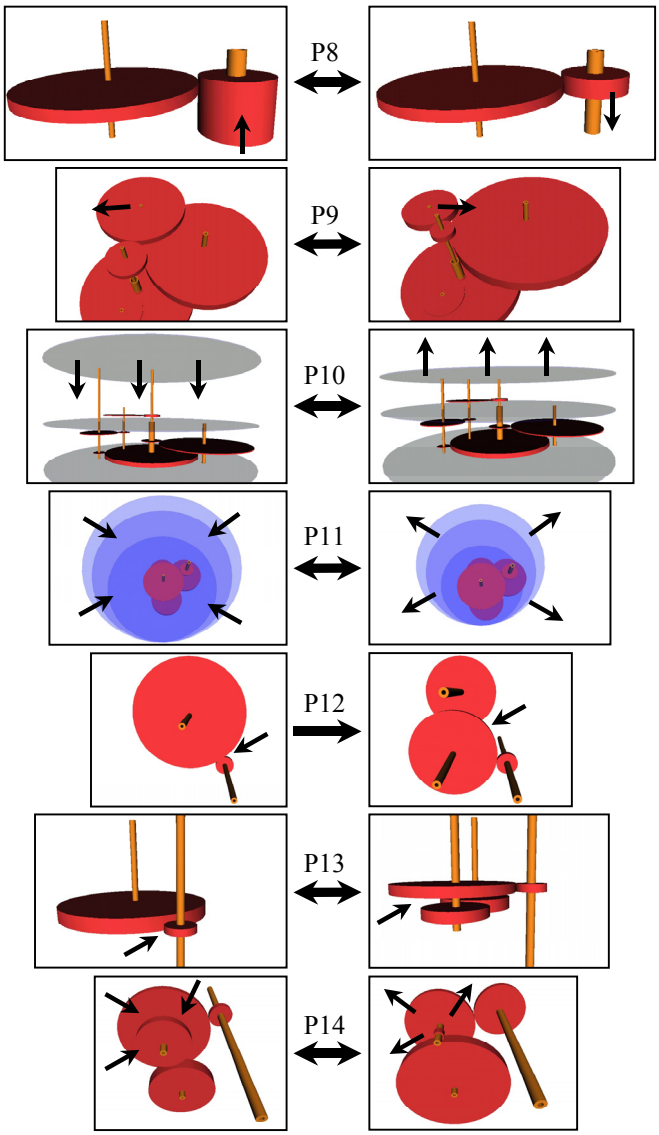

Figure 5-3: Perturb rules II

The perturb rules take a variety of parameters that enable their application point and resultant changes to be controlled. Of these, independent variables may be set by the user. As an example, P1, the rule that allows the upper spindle axle to be shifted, takes the scalar value of the desired shift as its input. As there can be more than one spindle in a design, the rule also takes a spindle identifier as an input as well. More complicated P-Rules take more inputs, e.g. P9, the rule that enables a spindle to be moved in the plane orthogonal to its axis, requires two scalar inputs to designate the shift in two-dimensional space as well as a spindle identifier to select the desired point of application. Dependent parameters are determined internally by the perturb rules: in the case of P12, the rule that inserts a new spindle and corresponding gear disks

into a structure, the position of the new spindle can be calculated from the user-specified radius of the new gear disks and the constraint requiring preservation of the overall ratio of the gear pair connections.

Table 5-1: P-Rule details I

P-Rule	Action	Rationale
P1: Set spindle top	Adjusts position of top of spindle (and hence length) while keeping the bottom fixed	Allows spindles to be either lengthened to add more gear disks, or reduced in length to avoid taking up space that could be used more usefully by other elements. Key constraint: interact.
P2: Set spindle bottom	Adjusts position of bottom of spindle (and hence length) while keeping the top fixed	
P3: Move gears vertically	Moves a gear pair in the axial direction	Allows variation of gear positions, for example allowing torsion to be reduced if two sets of gear pairs are too far apart. Key constraint: interact.
P4: Move spindle angle	Rotates a free spindle about its connecting spindle	Allows for rotation of free spindle to find a preferred valid location before connecting it to further spindles via new gear pairs. Key constraint: mesh.
P5: Set spindle r_out	Sets outer radius value of a spindle	Allows the creation of a shell to provide concentric spindles. Can be used to create stiffer spindles or reduce mass of a spindle.
P6: Set spindle r_in	Sets inner radius value of a spindle	
P7: Set gear top	Adjusts position of top of gear disk	Changes contact area of gear disks which might be required by a particular gear mechanism. Key constraint: interact.
P8: Set gear bottom	Adjusts position of bottom of gear disk	

As with existing parallel grammar rules, these new perturb rules are not meant to be an exclusive set. Designed with flexibility in mind, they are intended to be as generic as possible so as to encourage compatibility with other design domains.

Table 5-2: P-Rule details II

P-Rule	Action	Rationale
P9: Move spindle	Moves spindle in plane normal to axial direction	Allows rearrangement of spindles to create more favourable distribution of gear sizes, e.g. if a reduced aspect ratio of gear disks is required. Key constraints: ratio, mesh.
P10: Move plate	Moves a base plate in axial direction	Allows reduction in volume of overall clock. Allows more space to be created in clock structure if space is too crowded for further rule applications.
P11: Change plate radius	Adjusts radii of base plates	Allows reduction in volume of overall clock by reducing the radius of base plates. Allows more space to be created within clock structure if space is too crowded.
P12: Insert spindle P13: Delete spindle	Inserts a new spindle with gear pairs into the structure Deletes a spindle with its gears from a structure	Allows modification of designs between design families. These rules allow minor system topology to be changed, so increasing the possible design solutions that can be obtained by searching from any particular initial solution.
P14: Vary gear radius	Varies the ratio of neighbouring gear pairs	Allows variation of local balance between gear ratios. P12 inserts spindle with gear disks of equal radius, this rule can therefore alter this balance so that new spindles can be created with different sized gear disks.

An example of the use of perturb rules as part of the design generation process is shown as a series of stepwise rule applications shown in Figure 5-4. A partial design has been generated with concentric spindles at the origin (top left). A spindle is to be added and connected, via a new gear pair, to one of the inner concentric spindles. This, however, is not possible as the inner spindle is completely blocked by the outer spindle that acts as a protective sleeve. This situation can be addressed using two P-Rules. P10 moves the middle support plate down to make space (top right), followed by an application of P1 to reduce the height of the blocking spindle (bottom right). This enables the successful application of structure rules 2 and 3 to add a new spindle and gear pair (bottom left).

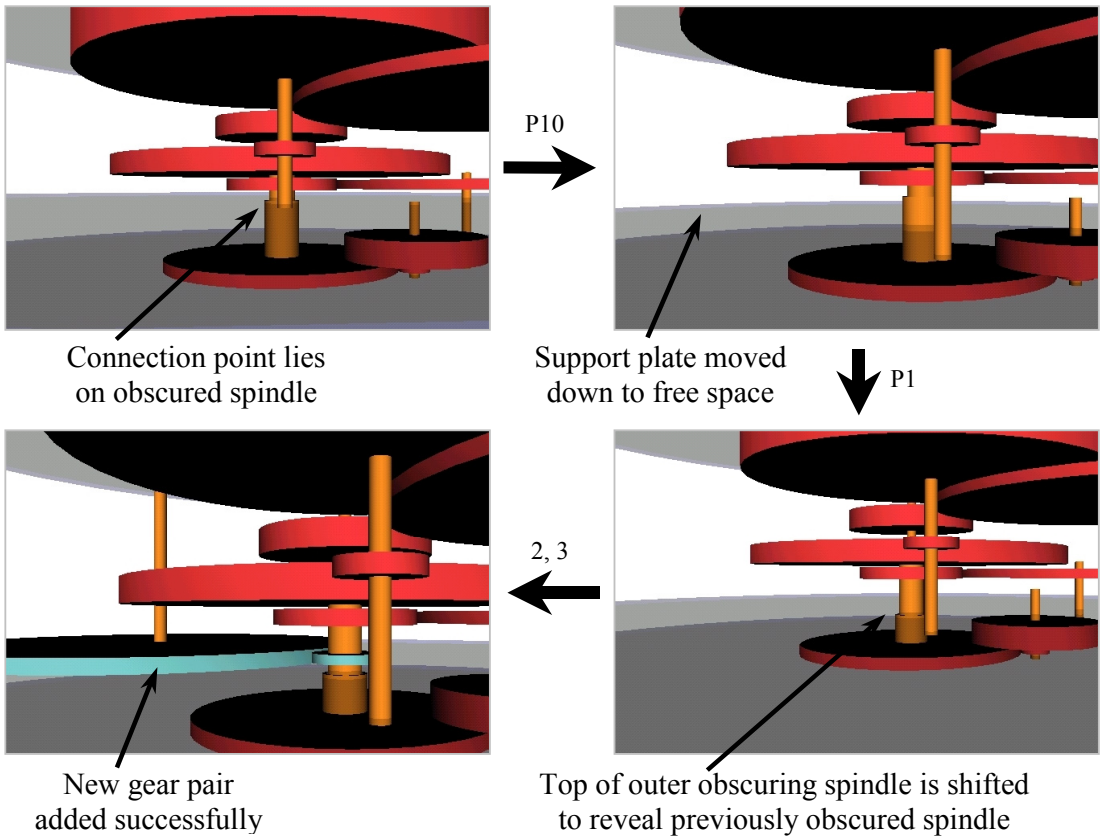

Figure 5-4: Application of P-Rules to enable continuation of design generation. P10 and P1 free up space on a previously obscured spindle to allow structure rules 2 and 3 to be applied to add a new spindle and connecting gear pair (light blue disks in bottom left picture).

A final point to note is that the perturb rules do not affect the system topology of designs, i.e. any alterations result in designs that are within the same clan. Perturb rules P12 and P13 do alter the design family, thereby resulting in a change to the function graph of the design. For example, applying P12 to insert a new spindle in a gear train would result in a new node in the function graph as well. This change is captured by a new function rule, function rule D, shown in Figure 5-5. A left-to-right application of rule D corresponds to perturb rule P12, a right-to-left application of rule D corresponds to perturb rule P13. The set of grammar rules used for a particular design problem can be selected from the total library of grammar rules available. If node insertion and deletion is not appropriate for the design problem under investigation, then the grammar rules that enable these changes are not used.

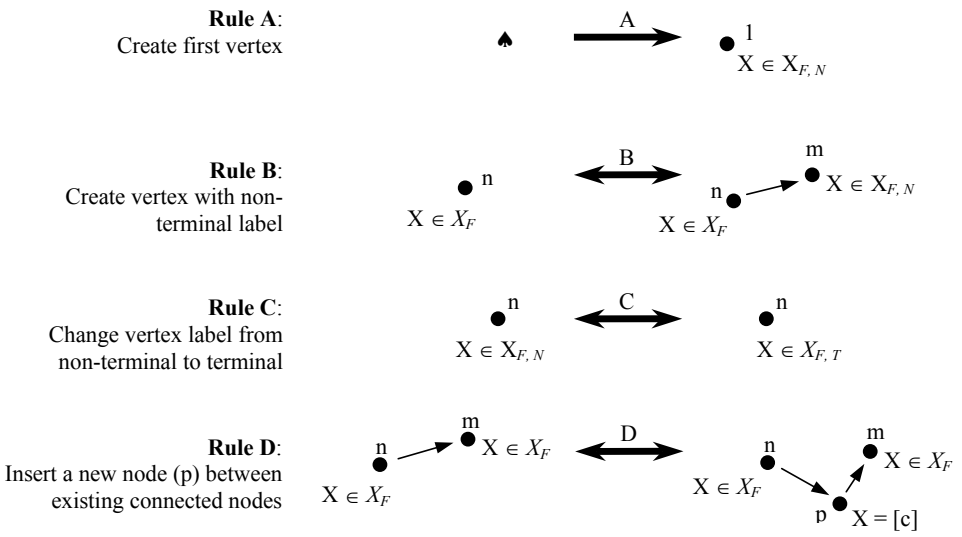

Figure 5-5: Set of function rules R_F with new rule D to insert/delete nodes

5.2. *Geometric design metrics*

The perturb rules introduced in section 5.1 represent one part of the two-pronged strategy for improving design generation as they introduce the possibility of modifying designs while simultaneously upholding their functionality. Having introduced the machinery to enable such design modification, a method of making decisions about the quality of designs is required to complete the second part of the strategy for design generation improvement.

As discussed in chapter 3, determining the quality of mechanical designs is no easy task. Papalambros and Wilde (2000) go so far as to say that a 'criterion for evaluating alternatives and choosing the "best" one cannot be unique'. Informative judgement on design performance can be ascertained with detailed modelling and analysis of virtual prototypes. The approach taken in this work is to look at geometric and behavioural performance, enabling a supply of knowledge about the predicted behaviour of the final product once it has been instantiated in an analysis model. However, behavioural simulation methods are often computationally intensive and are therefore not particularly suited to heuristic search methods that require a large number of iterations before they can be expected to find optimally directed designs (see section 3.6). Therefore, as an initial path of investigation, the possibility of using more time efficient evaluation methods is investigated.

The use of performance indices, or metrics, is prevalent in materials selection for design, e.g. (Ashby 1992). From this standpoint a series of geometry-based metrics were considered for assessing and comparing designs. The metrics are listed in Table 5-3 with explanatory text and a short rationale for each. These were not all produced at once and represent a subset of possible design metrics that could be used to assess mechanical systems. Some, for example the cost and mass metrics, are common from work in structural synthesis. Others are more specific to mechanical design, such as the compactness metric that provides a measure of both total and component volume. Two of the metrics in this list introduce design-specific weightings that can be used to fine-tune the performance of search strategies in a manner that will be discussed in more detail later in this chapter.

One aspect that all the design metrics in Table 5-3 have in common is that their values for a particular design can be calculated from the variables of the structure representation. Hence a complete or partial design can be analysed rapidly using these design metrics and the resulting information can be made available to a search process wishing to make a decision on whether to accept or reject a particular parametric design change. The design metrics in Table 5-3 are used in this and subsequent chapters of this thesis.

Table 5-3: Geometry-based metrics

Metric	Description	Rationale
Simple volume V_{total}	Volume of complete structure as cylindrical bounding box.	Volume can be used as a measure of size of design. Designs with limiting spatial restrictions can be harder to generate as these can restrict design generation.
Simple compactness $V_{total} \cdot \sum_{n=1}^{N} V_n$	Product of the sum of all component volumes and total enclosed volume of complete structure.	Similar to simple volume metric but places a premium on unused space in a design.
Simple thickness $T_{highest} - T_{lowest}$	Thickness of complete structure.	Mechanical designs with low thickness can be desired. For example watches are required to be thin to fit discreetly on a user's wrist.
Aspect ratio $\sqrt{\sum_{n=1}^{N} \left({p_n}^2 \right)}$	Squared penalisation function of high aspect ratio components. Ratio p_n is quotient of major dimensions of each part. $p_n \geq 1$.	Load-bearing components with a high aspect ratio have longer, indirect force paths[32]. Additionally, injection moulded plastics are difficult to manufacture for high aspect ratios.
Weighted thickness $\sum_{n=1}^{N} (z_n - z_0) \cdot t_n$	Local thickness metric weighted by component distance from a chosen reference point.	An extension of the above simple thickness metric using a weighting that, when used with some search methods, encourages rule applications that can indirectly result in a reduction in thickness
Weighted volume $\sum_{n=1}^{N} (z_n - z_0) \cdot V_n$	Component volumes weighted with distance from chosen reference point.	An extension of the above simple volume metric using a weighting that, when used with some search methods, encourages rule applications that can indirectly result in a reduction in compactness.
Cost of materials $\sum_{n=1}^{N} c_n \cdot V_n$	Sum of products of cost of materials per unit volume data array.	Absolute financial cost of a design can be a limiting factor in some circumstances.
Mass of components $\sum_{n=1}^{N} \rho_n \cdot V_n$	Sum of products of component density and volume	Reducing (or increasing) mass can be of importance for some design situations.

[32] See (French 1999).

5.3. Searching for preferred designs

The task of finding preferred designs within the language described by the grammar falls into the category of non-linear constrained optimisation problems. The search space, i.e. the entire language defined by the grammar, is multimodal, non-convex and discontinuous due to the type of objective functions used and the coupled nature of mechanical products. Deterministic methods, such as algorithms that rely on gradient calculation, are not ideally suited to solving such problems.

The alternative to using deterministic methods is to employ a non-deterministic approach. This may be imagined as navigating through a search tree at random (section 3.6), i.e. examining nodes at random to find the best ones. Augmenting such direct search methods with rules-of-thumb, otherwise known as heuristics, can increase search efficiency (Winston 1993). Two relatively straightforward search methods, random downhill search and simulated annealing, are chosen for their simplicity and proven robustness for an initial test of the synthesis framework. The aim is to find preferred solutions as determined by the design metrics in Table 5-3.

5.3.1. Random downhill search

Random downhill (RD) search is a kind of depth-first tree-traversal search where new designs are generated at random. A path in a tree of nodes, i.e. solutions, is chosen randomly where each of these new nodes is a child of the previous node. This method ignores other nodes at the same depth and so the method plunges into the tree rather than exploring solutions at a particular depth as would be undertaken by a breadth-first approach. Introducing quality measurements allows a choice of paths to allow movement only to nodes that improve the calibre of the design states as perceived by design metrics.

With regard to employing perturb rules for a simple downhill search, any rule application that leads to an inferior design (in this case an increase in objective function if a minimisation of a design metric is desired) is rejected. Therefore simple

downhill search is monotonic in metric space[33], implying a certain naïvety: the heuristic used does not necessarily hold in the general case, i.e. the best route to the global solution may not necessarily, and very often does not, lie along the path determined by downhill search. This is shown in one-dimensional simple schematic form in Figure 5-6. Starting from design (1), the random downhill search algorithm can find the preferred design state (2). Having located this local optimum the algorithm is unable to climb out of this valley to continue the search and make progress towards more optimal designs, such as design state (3). Complex designs have large numbers of variables, resulting in a multi-dimensional terrain with many local optima that slow progress towards optimally directed designs.

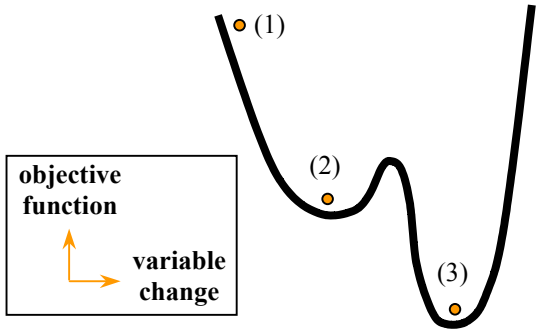

Figure 5-6: Objective function values for (1) non-optimal, (2) locally optimal and (3) globally optimal design states
(globality refers to points within the bounds of this sketch)

Locating a global optimum can be a difficult task. Generating a new best design disproves the globality of the previous best solution. However, this does not prove that this newest solution is a global optimum. In cases where an active constraint has been reached, such as a minimum thickness, then global optimality can be ensured, though in most cases search progress will cease before this theoretical limit is reached. It is then difficult to say with certainty whether or not a more optimal solution could be found by search along a different path.

The RD algorithm is shown in Figure 5-7. User-defined parameters for P-Rules are selected at random as part of the search algorithm.

[33] i.e. the algorithm is monotonous in its quest for improved designs. Design changes that do not improve the objective function are rejected.

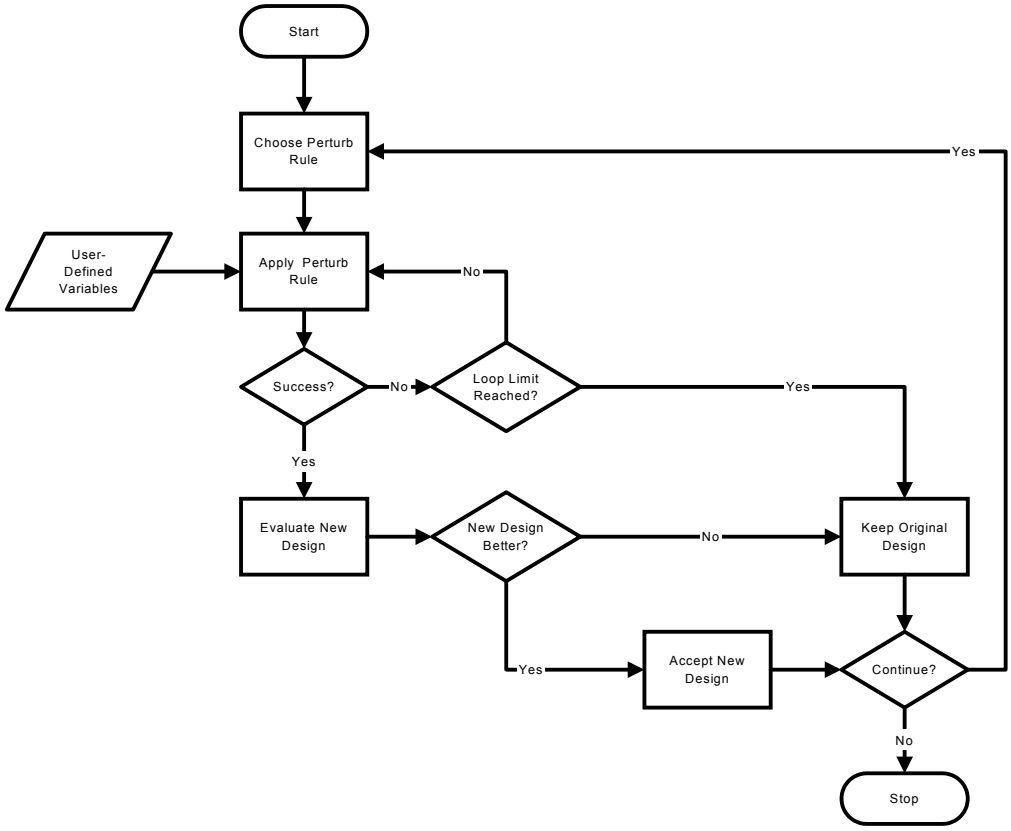

Figure 5-7: Random downhill search algorithm

5.3.2. Simulated annealing

Simulated annealing (SA) is a stochastic search method that is analogous to a type of heat treatment process for metal alloys. Annealing refers to the gentle warming of an alloy to a temperature sufficient to encourage diffusion, a thermally activated process, without actually melting the material[34]. This annealing process removes microstructure artefacts such as dislocations that may have been introduced through processes such as cold-working or irradiation. The annealing process is therefore a homogenisation process. As an example, some types of nuclear reactor vessels require periodic annealing to restore their original structural properties after continuous bombardment by neutrons has resulted in a destabilised microstructure.

[34] A common rule-of-thumb states that the annealing temperature for an alloy is $0.6 \cdot T_m$, where T_m is the melting temperature of that alloy (Ashby and Jones 1998).

The analogy with heat treatment of metals must not be taken too far. However, the concept of post-anneal cooling of a metal is a good way of understanding how the simulated annealing search process works. Having 'heated' (homogenised) the design for a period of time to allow perturbations that may result in a less optimal value of the objective function, a period of 'cooling' follows whereby the 'temperature' of the design is reduced. As this happens, changes are increasingly likely only to be accepted if they result in an improvement of the design. The 'freezing' process eventually fixes the design in a preferred state. The SA algorithm as implemented for use with the parallel grammar is shown in flowchart form in Figure 5-8.

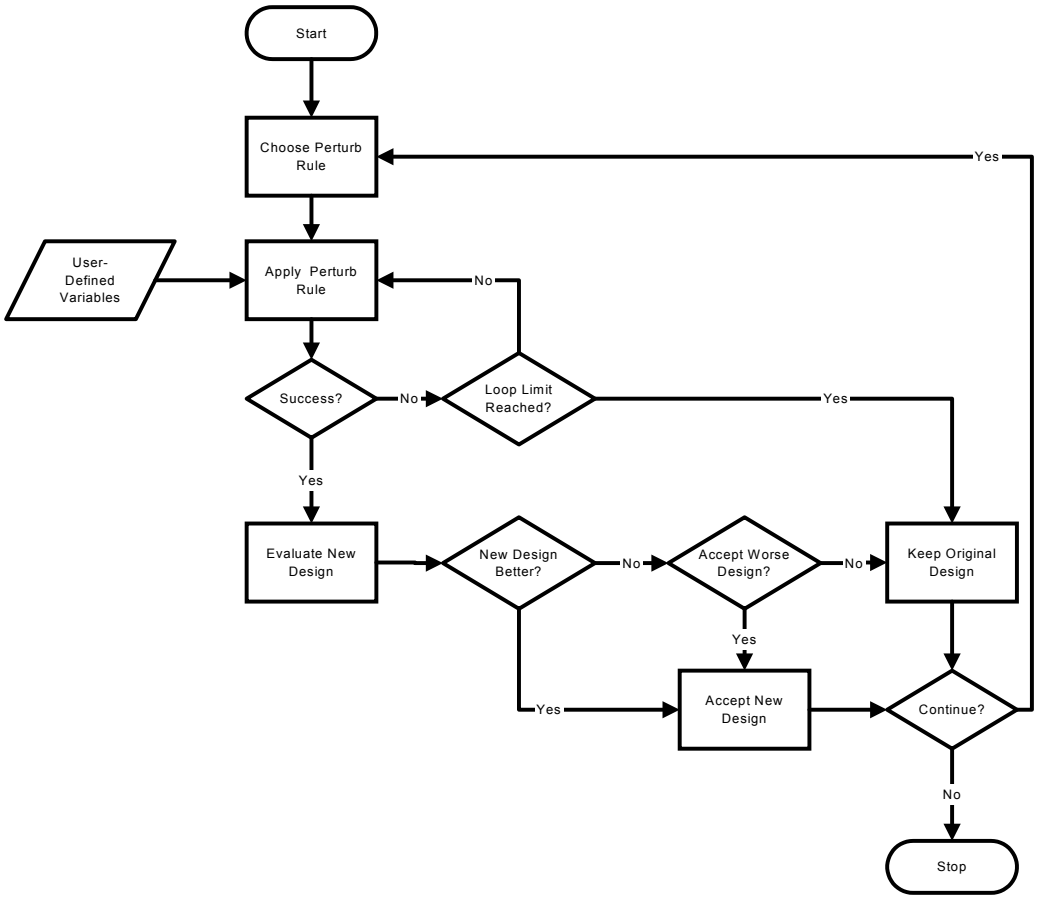

Figure 5-8: Simulated annealing search algorithm

SA search is non-monotonic in metric space as it allows steps from 'better' to 'worse' designs. This allows the search method to avoid entrapment in local minima that are far removed from more optimally directed designs (see Figure 5-6). As the temperature parameter is reduced, the likelihood of such events occurring is reduced. Additionally the step length (analogous to the diffusion distance in metals) is lessened

to encourage more local exploration around the current design state. The parameters that control these aspects of a typical simulated annealing process are listed in Table 5-4. The 'annealing schedule' is the term used to describe temperature as a function of design iterations. Many different schedules exist, e.g. (Lam and Delosme 1998). The common 'vanilla' schedule is used here where the temperature is reduced by a constant factor after a fixed number of attempted or successful design changes. Simulated annealing has been described as a 'time-consuming, iteration-intensive algorithm' (Szykman et al. 1997) as search iterations carried out at annealing temperatures can use up considerable computational resources.

Table 5-4: List of fundamental SA parameters with common value ranges used

Parameter	Description
$N_{\max}$	Number of iterations, i.e. design perturbations, allowed before the search must terminate. Not strictly part of the simulated annealing algorithm, it gives an upper bound on the amount of calculations performed and therefore controls the extent to which search is continued. The efficiencies of calculations with equal values for this parameter may be compared. Common value range: 10000 – 100000.
$\mu_{\min}$	Number of accepted design changes that trigger a drop in temperature. This parameter is reset to zero if a separate event causes a temperature drop. Common value range: 200 – 400.
L_k	Length of Markov chain (in this case, number of attempted design changes) that triggers a drop in temperature. This parameter is reset to zero if a separate event causes a temperature drop. Common value range: 500 – 1000.
T_0	The starting temperature of the algorithm. The temperate T is reduced from T_0 as the search progresses. T controls the probability of accepting a design change that results in a design state that is less preferred than the original state. Common starting value range: 1 – 10.
α	The factor α controls the temperature drop for the vanilla annealing schedule. Common value range: 0.0001 – 0.005.

5.3.3. Random downhill search vs. simulated annealing

The main subject matter of interest in this chapter is the investigation into the feasibility of generating viable mechanical designs using an extended parallel grammar that incorporates perturb rules. The search methods were chosen to enable a robust proof of concept for the synthesis framework. The random downhill search method was adopted for its ease of use and conceptual simplicity. Simulated annealing was chosen as a proven, robust and mature method that has been used successfully in synthesis research, e.g. (Szykman and Cagan 1997). More complicated to implement than the random downhill search, it is nevertheless suitable for use with modification rules (i.e. P-Rules) with relatively few changes from a standard implementation as might be used for parametric search rather than synthesis. Strengths and weaknesses of the two methods used have been summarised in Table 5-5.

Table 5-5: Summary of search methods used

Search method	Advantages	Disadvantages
Random downhill (RD)	• Straightforward to implement	• Monotonic in metric space
Simulated annealing (SA)	• Robust • Non-monotonic in metric space	• Performance dependent on well-chosen parameters

5.4. Improved design generation

An initial detailed comparative search was carried out for both the RD and SA algorithms. The specification graph used for this comparison is the same as that used for the most complicated design problem in chapter 4 (Figure 4-18). The initial solutions generated for this design (e.g. Figure 5-9) are not optimal due to the large spacing between components in the design. Using perturb rules as part of a search algorithm allows preferred designs to be generated from this, and other, initial designs.

The two algorithms were compared by initially generating five separate starting designs for a bounding cylinder of radius 35 mm and height 20 mm. The second

initial solution of these five is shown in Figure 5-9. These initial designs, the same as the valid designs generated in chapter 4, were each used as the 'seed' for 10 separate search processes, or experiments, using both random downhill and simulated annealing search algorithms to give a total of 50 final solutions for each search method. A reduction in the thickness of designs was used as the sole objective function for the search process.

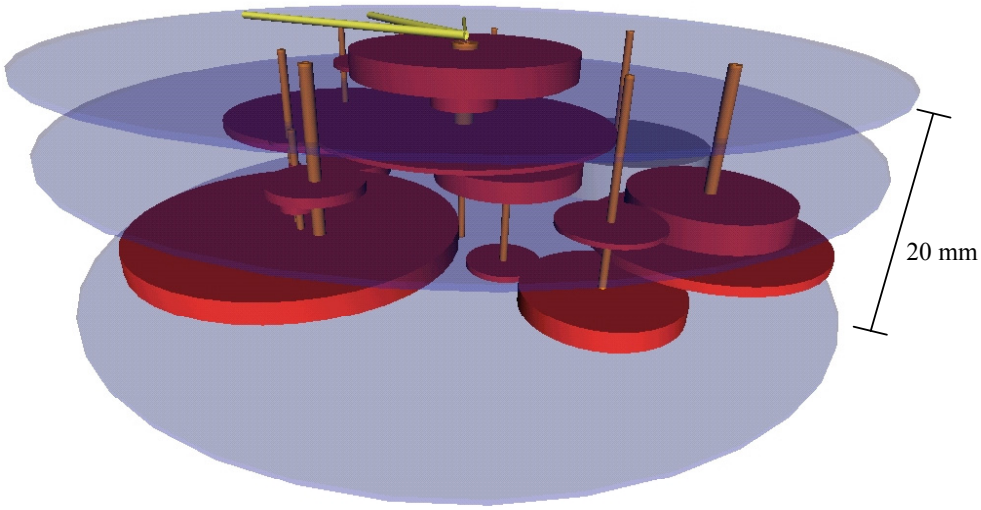

Figure 5-9: Initial solution #2. Thickness of initial solution is 20 mm.

A relatively large number of iterations were selected for each problem ($N_{max} = 50{,}000$) to ensure an adequate capture of convergence behaviour. Simulated annealing parameters were chosen after a few initial investigative trials to provide sensible values in line with common practice ($\mu_{min} = 400$, $L_k = 1000$, $T_0 = 2$, $\alpha = 0.0001$). A main issue prior to the use of perturb rules was that generated designs tended towards wasteful packing of components, resulting in large structures with a large distance between the top and bottom of the design, i.e. a large thickness. In general, thinner clocks are desirable. Consider as an example a wall-hanging pendulum clock. Reducing the radius may not be of such great consideration as this reduces the size (and therefore the visibility) of the clock face, however, a thinner structure leads to a design with mass concentrated closer to the wall attachment, so reducing the tension on the attachment that keeps it in place on the wall.

Both the random downhill and simulated annealing algorithms were able to modify the five initial designs successfully and generate thin designs. The average values of thickness metric over the 50 designs were very similar at just over 8 mm. The best values achieved by the two algorithms differed by a greater margin: the best SA solution was 5.7 mm, 1.0 mm thinner than the best RD solution. A summary of this data set (solution set 5A) is shown in Table 5-6.

Table 5-6: Summary of random downhill and simulated annealing solutions for thin clock designs – data set 5A

Thickness [mm]	**RD**	**SA**
Average value	8.3	8.1
Minimum value	6.7	5.7
Maximum value	11.4	10.6
Standard deviation	1.1	1.1

The data in Table 5-6 confirm that there is no great difference between the RD and SA algorithms in this example and for these search parameter values. The SA algorithm, as expected, produces some better results due to its ability to extricate itself from local minima, however, the algorithm also spends a significant number of iterations at a higher temperature in which preferred designs are not found. This can be seen by comparing the progress of the two algorithms for one of the initial solutions in data set 5A (Figure 5-10).

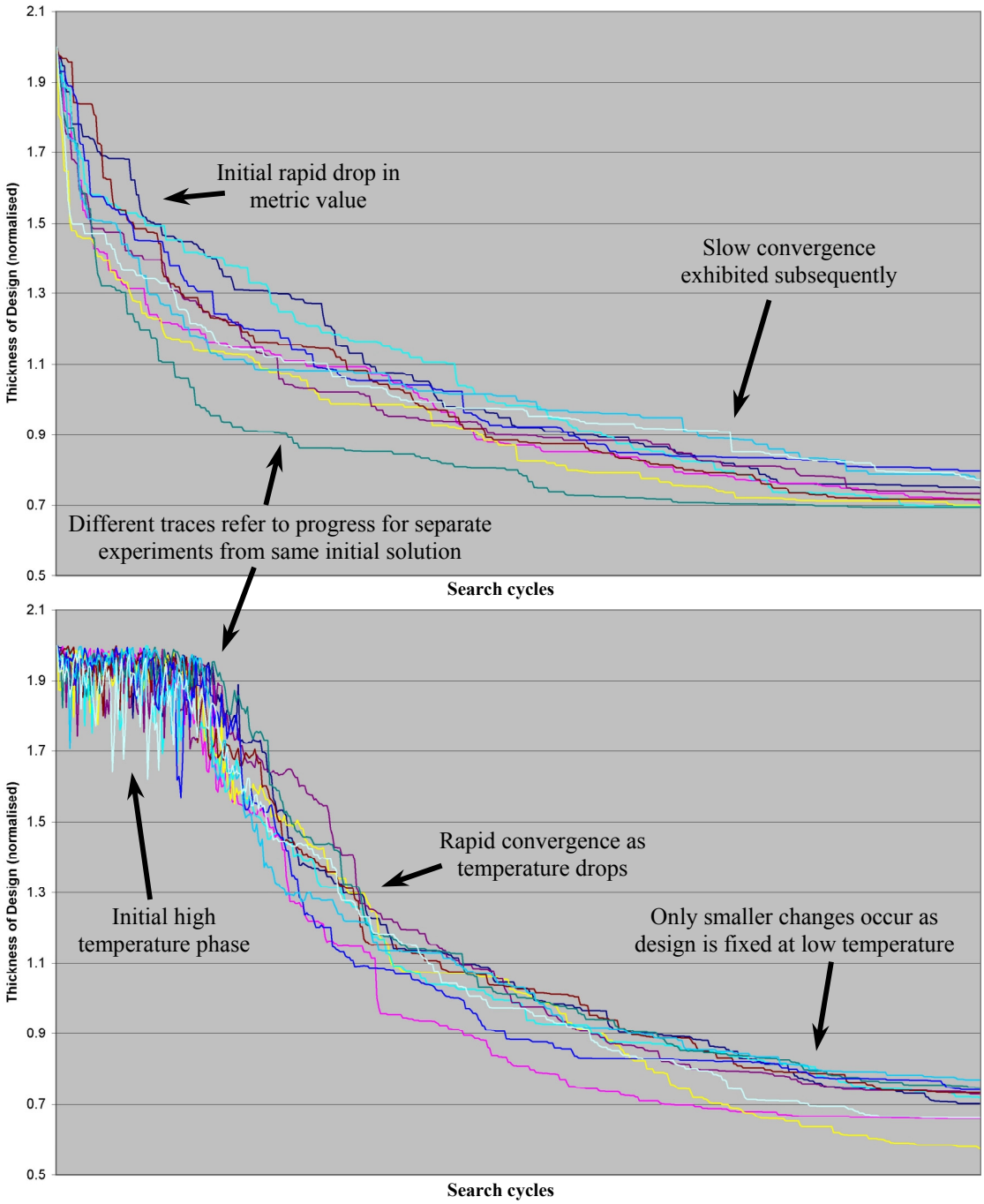

Figure 5-10: Progress of RD (top) and SA (bottom) algorithms over 50,000 iterations. Solution set 5A, initial solution #2, progress traces shown for each of 10 experiments.

Figure 5-10 shows the progress traces for 10 experiments starting from the initial solution in Figure 5-9. The traces are compatible with expected performance of the algorithms. The RD algorithm makes initial rapid progress but is undermined by the simplicity of its selection process. The SA algorithm makes no progress until the temperature is dropped. However, it then makes more rapid moves towards preferred

designs. The traces flatten out as the temperature of the experiment is reduced and, therefore, the probability of accepting designs with a higher metric value drop to zero.

The best design generated by the SA algorithm is depicted in Figure 5-11. With a thickness of only 5.7 mm it is on the boundaries of feasibility for a clockwork mechanism. The absolute theoretical minimum for this particular design is slightly greater than 5.2 mm. This is calculated as the sum of the required minimum thickness of the clockwork power source, the thickness of the three plates that the spindles are connected to and a small stand-off distance between each component. The evaluation procedure used for generating this design only considers overall thickness and has no means of recognising the value of attempting to generate more compact components that in turn will enable the generation of designs closer to the theoretical minimum. It is therefore highly unlikely that a complex function graph such as that in Figure 4-18 could be instantiated as a valid structure at the minimum thickness without considering more than merely the overall distance between top and bottom support plates.

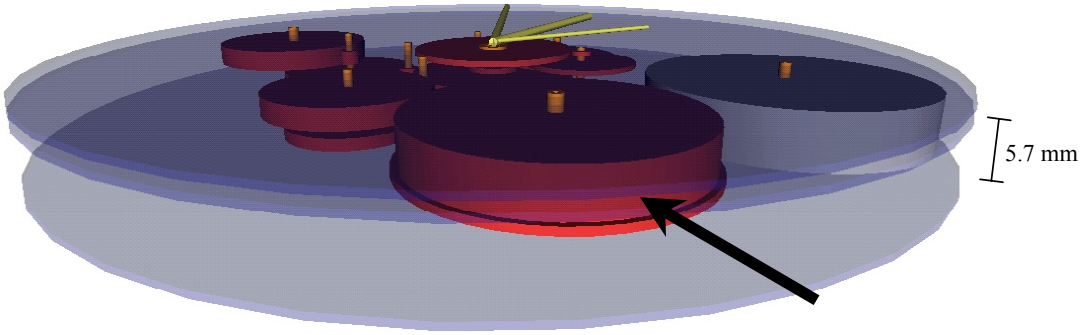

Figure 5-11: Data set 5A, simulated annealing algorithm solution #18 – thinnest design. Arrow highlights example of chunky gear disk.

The hypothesis that simple design metrics may be used to drive the design of complicated designs, as introduced in section 5.2, begins to show some failings here. Inspection of the design in Figure 5-11 brings to light some gear disks that are fairly chunky, thicker than the minimum thickness requirement (determined by the spatial constraint). Reducing these in size might free up more space in the centre of the structure. P-Rule applications may then be able to increase the compactness of some of the components as mentioned above. However, the simple thickness metric does

not provide a drive for the simple simulated annealing algorithm to reduce the thickness of individual components.

It is required to reassess the evaluation method used in order to achieve this space reduction. Table 5-3 provides us with a battery of metrics for design evaluation. So far just one has been used as the driving force for searching out preferred designs. Rather than considering overall design thickness, using a weighted thickness metric should result in greater consideration of individual components in the design. The weighted thickness metric considers the centre of mass of each design component and multiplies it by its distance from a user-defined point, in this case the clock face (Table 5-3). This should drive both the individual thickness of components down while in turn also reducing the overall thickness of the whole design by freeing space in the centre of the design to allow the outer support plates to be moved closer together.

The same five initial designs as above were used in an identical experiment to produce 50 design solutions using the weighted thickness metric. Again, both RD and SA algorithms were used to generate solutions. The metric was a successful evaluation method for the two search methods, allowing both to generate good solutions as tabulated below (see Table 5-7 for this data set 5B). The chunky gear disks from before do not get created, a result of using the modified design metric for evaluation. The overall aim, however, is still to reduce the overall thickness of designs. With respect to the simple thickness metric the results are not as good as those generated previously. How can this be? Inspection of the resulting designs again provides an answer. The mapping between the two metrics is fairly good, but cannot be perfect. For example, the best design produced using the simple thickness metric alone scores highest on the weighted thickness metric scale. However, this is not always the case as the emphasis introduced by the weighted metric dilutes the pure drive to produce an overall thin design.

Table 5-7: Results for weighted thickness metric – data set 5B

	RD algorithm		**SA algorithm**	
	Weighted thickness [normalised]	Thickness [mm]	Weighted thickness [normalised]	Thickness [mm]
Average	0.45	15.3	0.38	13.8
Minimum	0.34	12.1	0.26	9.4
Maximum	0.64	18.0	0.54	17.6
Standard deviation	0.09	1.5	0.07	2.2

It seems that the ideal solution may be found using a combination of objective functions. Merely considering the simple thickness metric results in thin designs that require small improvements (reduction in size of the chunky gear disks) that are obvious to a designer upon inspection of the results. Introducing a new metric, the weighted thickness metric, to address these issues results in an eradication of the problematic chunky gear disks but softens the focus on producing thin designs as originally specified.

To generate designs that balance the two goals, a linear combination of the simple and weighted thickness metrics was used for the same design problem as before. Four separate SA search runs were carried out with ratios of simple to weighted thickness metric of 8:1, 4:1, 2:1 and 1:1 respectively. These ratios were applied after metric normalisation. As usual, the same five initial designs were used to generate 50 new design solutions. The results are reproduced in Table 5-8.

Table 5-8: Results for combined simple and weighted thickness metrics – data set 5C

Thickness [mm]	Ratio 8:1	Ratio 4:1	Ratio 2:1	Ratio 1:1
Average	9.1	10.9	12.2	12.7
Minimum	6.4	6.8	8.0	7.6
Maximum	14.9	15.9	16.9	17.8
Standard deviation	2.1	2.7	2.2	2.2

This data shows a best performance for the high ratio (8:1) of simple to weighted thickness metric that is just short of the quality provided by the pure simple thickness metric (Table 5-6), the best result is a design of 6.4 mm thickness. Use of this

combined metric assists the finding of design solutions that are both thin overall and have thin components. These results therefore present a trade-off for the designer of solutions that satisfice good overall design criteria.

A similar approach was used for the generation of data set 5D (Table 5-9), replacing the weighted thickness metric with a simple mass metric as defined in Table 5-3. For the first time this provides a real drive to reduce the width of components as well as their thickness. This is because a mass reduction is forthcoming if the volume of materials is reduced. The results are slightly better than those of data set 5C, suggesting that a more targeted approach at aiming for an overall increase in compactness should also be successful at generating thin designs.

Table 5-9: Results for simple thickness and mass metrics – data set 5D

Thickness [mm]	Ratio 8:1	Ratio 4:1	Ratio 2:1	Ratio 1:1
Average	8.9	10.6	11.3	12.1
Minimum	6.2	6.9	7.9	7.2
Maximum	13.6	16.3	16.5	17.3
Standard deviation	2.0	2.5	2.3	2.4

The last experiment of this chapter was carried out using only the compactness metric from Table 5-3. Evaluation centres on the volume of individual components and that of the overall design with a reduction in either leading to designs that score well for this diagnostic. The results are better than for all previous experiments, both RD and SA algorithms achieving consistently thin designs with minimum values below the best previous design solutions. Surprisingly, the random downhill algorithm outscores its simulated annealing counterpart, presumably because the compactness metric is effective at successfully rewarding preferred design modifications and so negates the positive effects of the initial annealing phase on the success of the search.

Table 5-10: Results for search using compactness metric – data set 5E

	RD algorithm		**SA algorithm**	
	Compactness [compound value]	Thickness [mm]	Compactness [compound value]	Thickness [mm]
Average	0.22	7.5	0.27	8.0
Minimum	0.18	5.4	0.20	6.2
Maximum	0.36	10.0	0.35	10.0
Standard deviation	0.03	1.0	0.04	0.9

5.5. Conclusions

This chapter has extended the original parallel grammar by adding modification rules that allow perturbation of the structure representations of partial and complete designs. Using perturb rules, the existing design generation process has been improved so as to enable parametric exploration of a design space. Coupled with simple metrics for design evaluation, this has allowed the use of non-deterministic search algorithms for computational generation of preferred designs. Metrics, based on geometric properties such as mass, thickness and compactness, were successfully used for generation of preferred designs using random downhill and simulated annealing search algorithms. The performance of different metrics was investigated by using them to drive search from an identical set of initial designs.

5.6. Limitations

The use of design metrics based on geometry alone allows for initial investigation of possible preferred designs using appropriate search algorithms, as these metrics provide a measure of performance in a manner that is efficient for initial study. For the simple design case study presented in this section, these metrics prove adequate to an extent, enabling the generation of preferred designs for constrained packing problems, e.g. (Szykman and Cagan 1997), with parametric components and part connectivity constraints. However, the actual behaviour of the generated designs is only implied by the function and structure representations used to represent the clocks of the case study. This can be seen as a disadvantage, as the possibility of computationally generating a palette of design possibilities for consideration loses its

appeal if potential designs still require verification by way of behavioural modelling. In addition, if this behavioural data has not been used during the generation process, one cannot be certain that final designs generated using simple geometric design metrics meet quantitative behavioural performance requirements. The aim of this research, after all, is to investigate the possibilities for performance-based and optimally directed synthesis of mechanical designs.

The idea of using automatic modelling and analysis of designs generated by computational methods must therefore be investigated in more detail. This, in turn, may well require a reassessment of the search algorithms used for design generation. Despite their robustness and ease of implementation, both the random downhill and simulated annealing algorithms used are relatively naïve and profligate (see section 5.3.3). Even if it were to prove possible to generate real-time analysis data, this is likely to come at a computational price and a more efficient search algorithm would be highly desirable.

A more efficient algorithm might benefit from using metric data more productively. Combining the weighted and simple thickness metrics to compose a successful single-valued function for more than one objective proved difficult, requiring repeated experiments using many different weighted combinations of metrics to provide good search results. While successful, the method is wasteful of both user time and search cycles. Additionally, the method is not conducive to clear presentation of search results.

In summary, the following is required:

- Automatic behavioural modelling during design generation to enhance the model used to drive synthesis.

- A search algorithm that is compatible with the parallel grammar and can locate preferred designs at less computational cost, i.e. in less time, than the RD and SA algorithms.

- Improved processing and presentation of performance data using true multi-objective search.

These three issues are addressed in chapter 6 of this thesis.

5.7. Implementation details

The original C++ class structure was extended to implement the work presented in this chapter. Both the perturb rules and design metrics were written as C++ functions of the main class used for the structure representation of the parallel grammar. New functions were verified by repeated application to simple problems to resolve inconsistencies. The user's selection of design evaluation metrics takes place through file input to avoid unnecessary recompilation of computer code.

Having the complete program implemented within one piece of modular code enabled the final compiled program to run rapidly. Having a low computational cost associated with design evaluation allowed long search runs to be carried out, each with many iterations, in a relatively short period of time. For the experiments described above, finding solutions for 50,000 iterations took of the order of minutes per solution: one particular measured example required 102 s on a 1200 MHz CPU.

6. Enhancing design evaluation

In this section the design evaluation phase of the parametric synthesis framework is investigated in more detail. The use of behavioural modelling as an evaluation method is considered as an extension to the geometry-based metrics of chapter 5. An electromechanical camera is examined using the behavioural modelling language Modelica. Behavioural simulations are run for parametric models generated by the parallel grammar and this data is used for the performance-based evaluation of the generated camera designs.

Even using relatively fast computers, running one such simulation takes of the order of tens of seconds to complete, e.g. as an example, a simulation of the camera model introduced in this section was analysed in 14.4 s using a PC with an AMD Athlon 2600+ CPU. The simple search algorithms from the previous chapter require many objective function evaluations in order to obtain satisfactory levels of convergence, hence using the search algorithms previously proposed is not feasible. Two changes to the existing generative method are therefore introduced. Firstly, a hybrid pattern search algorithm replaces the existing random downhill and simulated annealing methods. This new algorithm attempts repeated patterns of modification rules that have previously led to successful design modifications. Secondly, a true multi-objective approach is employed to make better use of time intensive calculations employed during the search process, allowing generation of a Pareto set of non-dominated solutions, rather than a single solution, for further consideration by a designer.

6.1. Framework for enhanced design evaluation

Having provided preliminary investigations into computational design generation and modification in the last two chapters, this section centres on the evaluation phase of the proposed mechanical design synthesis framework. Previously, metrics have been used to enable fast evaluation of designs based on geometric considerations. Consider evaluating the Audi R8 transmission introduced in chapter 1. A performance-based evaluation may be of value in this particular instance. Design criteria for the gearbox may include, for example, its robustness with respect to harsh manoeuvring and high accelerations, as well as how much energy as a percentage of total input is lost to friction in the transmission. Access to accurate analysis data of this kind would give an insight into the predicted lifetime and safe operating limits of the gearbox.

While this evaluation process can be carried out as part of a manual iterative design process, it can also be incorporated into an automatic procedure. This can be seen as 'closing the loop' on performance-based design, i.e. linking the generation and simulation-based evaluation phases of design into a process that does not require manual intervention.

Simulation-based design evaluation is incorporated into the design synthesis framework in the same manner as geometry-based evaluation, c.f. Figure 5-1. The evaluation phase is carried out by translating the current design to a behavioural model in the native format of the simulation tool. The simulation is run and outputs quantitative information about the behaviour of the design. This evaluation data is then used to decide whether or not the new design is an improvement on previous designs.

Analysis tasks are routinely carried out as part of engineering design, e.g. as discussed in section 1.2 for the sports car transmission example. In these cases, the modelling process is a manual procedure. Considerable effort can be required to create a meaningful behavioural model and run successful simulations. While the modelling process is mainly done by hand, optimisation routines are sometimes used to find the best parameter values for particular design components.

There is, however, a difficulty with such an approach for the work that is required here. As the method carries out synthesis, fundamentally different designs are generated, i.e. designs can be from different design families and clans. As the variation between designs increases from geometric parameter variation and component topology through to system topology, so do the differences in a behavioural model. They move from minor variable modification, changes in part count and related connectivity through to variations in actual components present in the design. Simulation models that reflect these changes can become increasingly difficult to create computationally, automatically, for such wholescale changes. This is an important issue as the design synthesis framework is meant to actively explore as large a search space of potential designs as is possible. Limiting the search space to small parameter variation of a starting design would not allow performance-driven synthesis.

6.2. Behavioural modelling

The cross-domain object-oriented modelling language Modelica (c.f. footnote 14 on page 52) has been chosen for this work. Modelica models can be simulated using Dymola (c.f. footnote 15 on page 55), a simulation environment, to enable the prediction of dynamic behaviour and interaction between components in the design. A large range of component libraries are available that cover many different engineering domains, including mechanical systems. These basic building blocks can be used to create multi-domain design models, for example of mechatronic devices that combine mechanical components and electronic circuits. The general framework for simulation-based evaluation is not limited, however, to this particular behavioural modelling technique.

Figure 6-1 illustrates a Modelica class representation of the simple gear pair shown in Figure 4-9. This class was built up from library components such as inertia, shape and friction elements. It was created using a graphical user interface (GUI) provided by the Dymola environment that allows user-level interaction to choose, edit and combine the existing elements as well as construct new elements. This class structure can be built up to create complex interacting models using instantiations of a variety

of component elements. In this way it is possible to combine a number of such gear objects as well as similarly constructed elements for escapement and clockwork power source mechanisms to represent the clocks generated in the previous two chapters.

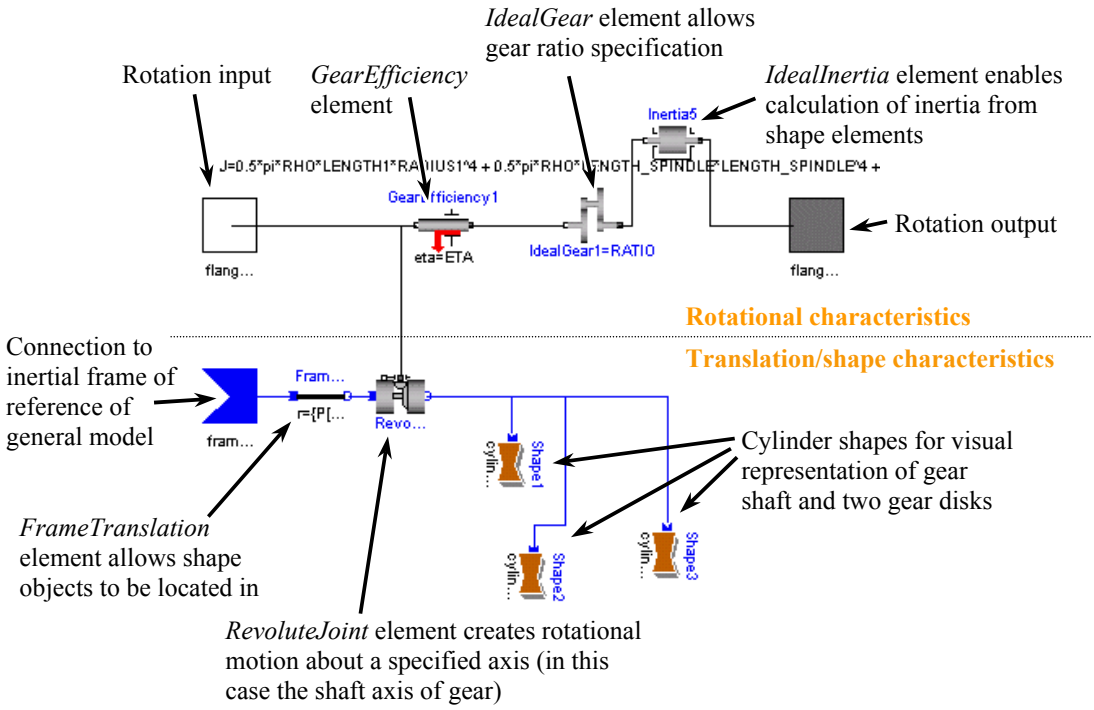

Figure 6-1: A simple Modelica model of a gear pair

6.2.1. Simulation

Having constructed a Modelica model, the Dymola environment enables the running of simulations. A simulation model consists of a compiled executable and an input file containing starting values of all parameters in the model. These files must be initially generated using the Dymola GUI interface. Running the simulation then calculates the behaviour of the system using Kron's method of tearing to solve the system of equations (Kron 1963). Upon completing the simulation the traces of each parameter in the model are written to files that can then be analysed using the Dymola environment or exported to other programs such as Matlab for further investigation.

This method of analysis is excellent for prototyping and manual testing of design concepts. It requires human intervention at each step of the process and is therefore

not appropriate for use as an evaluation method invoked by a search algorithm. Additionally, the GUI diverts computing resources away from the main program running the simulation, slowing the process down. It is imperative that the evaluation phase takes as little computation time as possible to ensure that generation of and search for beneficial designs proceeds as rapidly as possible.

Once the simulation executable has been compiled it can be re-used, separately from the Dymola GUI, as a standalone program. A new simulation can be prepared by updating the parameter values in the original parameter file to correspond to those of the newly created design. This updating process can be effected efficiently with Perl scripts to strip out and replace the original parameters in the input file. Subsequently running the simulation executable as a standalone program in combination with this altered input file enables fast, automatic analysis of a newly synthesised design.

6.2.2. Camera mechanism example

A fresh design task is introduced at this stage to outline the type of design synthesis issues that can be investigated using performance-based evaluation. The synthesis task involves the redesign of a mechatronic camera design based on an original product.

Camera design is an interesting example of how behavioural characteristics are an important aspect of design. Compactness is viewed as desirable in this design domain, but only without a degradation in behaviour, such as the quantity of light passing through the camera lens, battery consumption and, if the camera has an automatic winding feature to prepare the film for the exposure, the time taken for this winding process. A main feature of high performance sports camera equipment is the possibility of reeling off many frames per second to increase the likelihood of achieving a good picture of a fast action sequence. Naturally, the balance of requirements for different products and their target market varies. For single-use cameras, power consumption is not an issue as these cameras usually require manual winding and a battery can be made large enough to provide flash for each exposure.

The modelling aspect of this case study, initially undertaken as a diploma project in collaboration with Bolognini (2003), consists of representing and linking the winding and shutter mechanisms of the camera. A basic multi-use camera, a Vivitar CV50, was reverse-engineered to put together a Function-Behaviour-Structure representation (Figure 6-2). The camera is manually controlled and functions by allowing photos to be stored, frame by frame, through the brief opening of the camera shutter to expose sequential portions of the camera film. After each such exposure, the film is moved forward by the winding mechanism to prepare a fresh frame for exposure and therefore to ready the camera for the next photo. Until this action has occurred, a user cannot press the shutter release to instigate the taking of another photo.

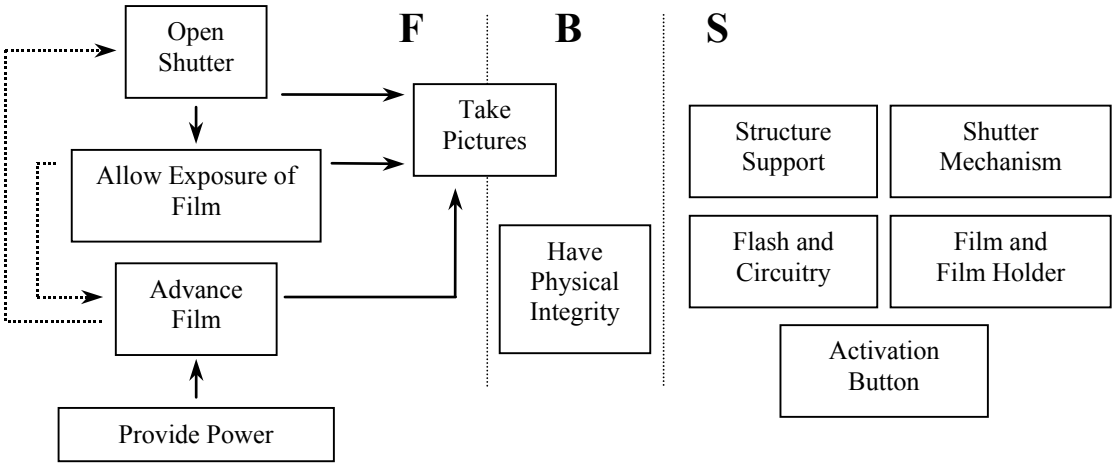

Figure 6-2: Function-Behaviour-Structure representation of a camera, adapted from (Bolognini 2003)

A performance analysis was carried out of the original camera and two manually redesigned models. Using the Dymola GUI, simulations were run to determine performance characteristics such as battery usage per winding cycle, time between pressing the shutter release and the actual photo being taken, and the time taken to wind on the camera for the next frame. Of the two new camera designs, one design was a parametrically modified version of the original model, i.e. it belonged to the same design family as the original design. The other design altered the part count in the gear train by removing a gear and manually changing parameters to ensure validity of the new construct. Due to the architecture changes this new design can be considered as being part of a new design family. This process was carried out by

hand, requiring considerable effort to set the correct parameters for the redesigned models (Bolognini 2003).

The question is now whether it is possible to use a computational synthesis approach to generate new camera designs. The manual redesign approach used to create and analyse the new camera designs is time-consuming and the number of possible alternatives is thus small. Can the behavioural feedback loop be closed to allow fast computational generation of new performance-driven designs?

The focus of investigation is the winding mechanism of the camera, essentially a linear gear train that transfers torque from the motor to the winding pivot at the base of the film cradle. These two items, film and motor, cannot be mounted in greater proximity to each other due to packing considerations in the main camera body. The current design, a linear gear train, is shown in Figure 6-3 with a close-up of the mechanism in Figure 6-4. There are seven spindles in total, the two ends of the gear train connected to the film and motor respectively as well as five gears associated with the gear train. These gears must sit in a confined space, precluding the possibility of using only a few larger sized gear disks to bridge the gap between film and motor as these disks would overlap with the camera casing.

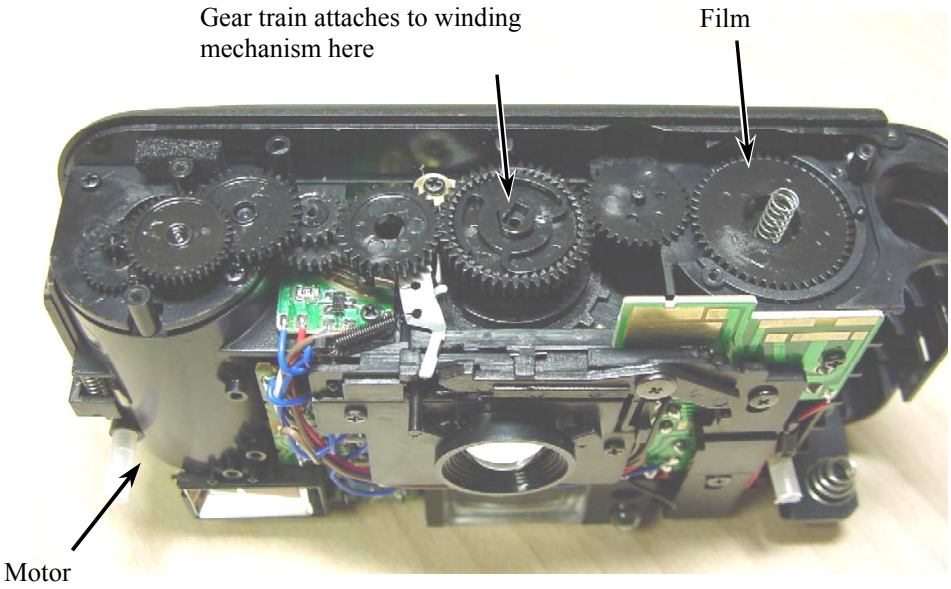

Figure 6-3: Vivitar camera with winding mechanism exposed (view from bottom)

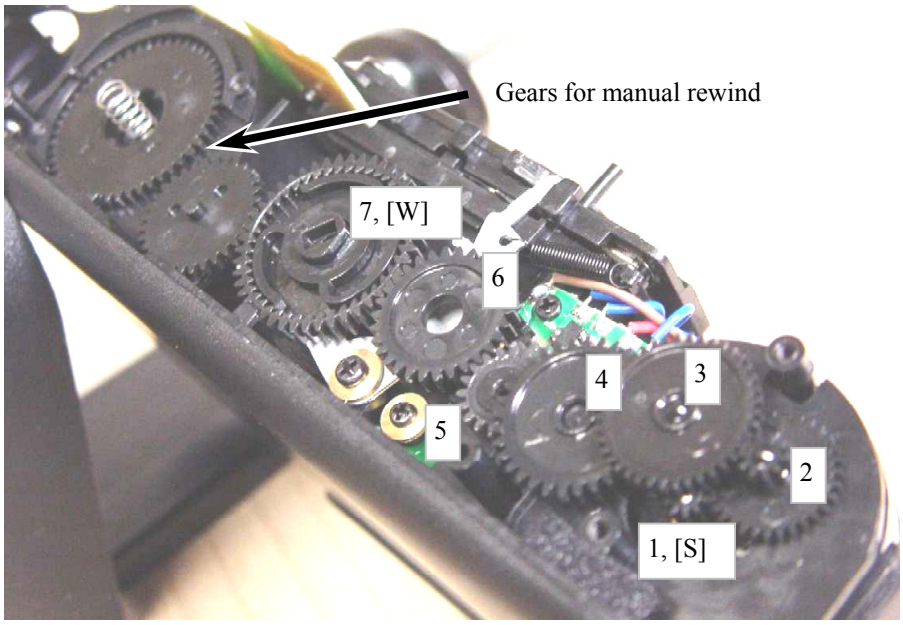

Figure 6-4: Winding mechanism of existing camera (gear train close-up)
Labels and numbering correspond to function graph in Figure 6-5.

Figure 6-5 shows the function graph of the winding mechanism (Figure 6-4). As it is a linear gear train, it is less complicated than some of the function graphs used to represent the clock designs previously. There are seven vertices in this function graph corresponding to the seven spindles in the gear train. These seven vertices are connected by a total of six edges; correspondingly, six separate gear pairs connect the spindles in the camera (c.f. Figure 6-4). The labels used for the camera model are listed in Table 6-1.

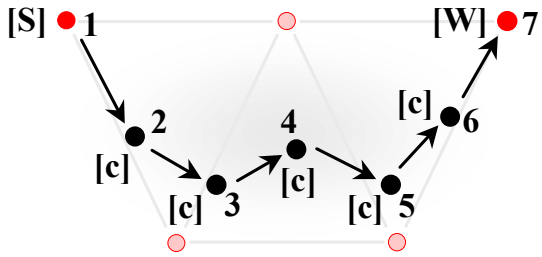

Figure 6-5: Function graph of original camera winding mechanism

Table 6-1: Explanation of terminal labels used for the camera design problem

Label	Meaning
c	A terminal vertex indicating further connectivity (no change from previous definition for clock grammar).
S	A terminal vertex that corresponds to the shaft of the motor in the camera design.
W	A terminal vertex that corresponds to the final axle of the winding mechanism that is used to wind on the film.

In order to enable compatibility with the existing parallel grammar and allow greater parameter variation, the different components in the winding mechanism (Bolognini 2003) were altered from the original Modelica specifications. The standard gear element is shown in Figure 6-1. The three shape elements that represent the gear disks and connecting spindle (bottom right of diagram) can be parametrically modified to vary the size and ratio of the gear. The inertia element of the gear class (top right of diagram) calculates the inertia of the element from these parameters. A gear train class is made up of a number of these gear elements. The existing gear train with five components is depicted in Figure 6-6. These five gear elements correspond to the five vertices in Figure 6-5 labelled as connecting vertices ([c]).

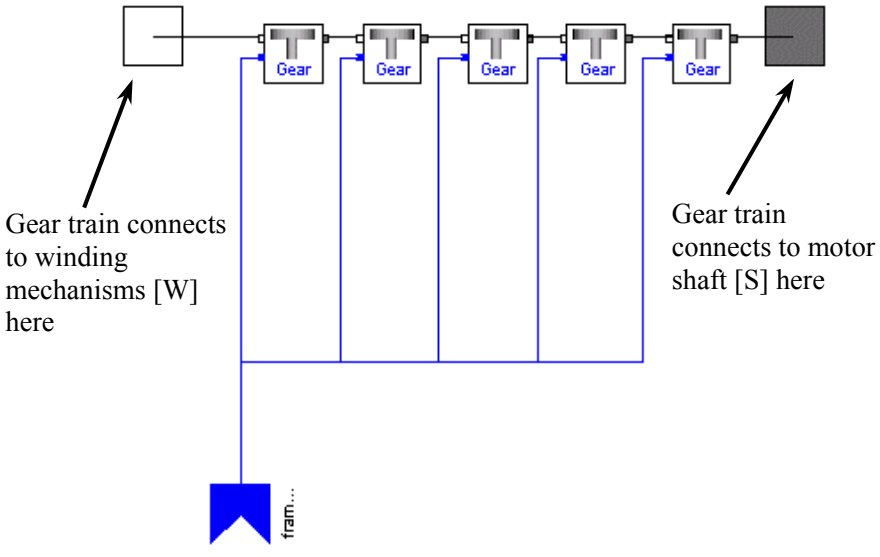

Figure 6-6: Gear train class (Bolognini 2003)

The two end spindles of the gear train (corresponding to vertices labelled [W] and [S] in the function graph in Figure 6-5) are fixed in space. These constraints are added as

user-defined general parameters that determine the position of spindles depending on their label. The end spindle that connects to the motor [S] is very similar to a standard gear element (Figure 6-1). At the other end of the gear train, the spindle that is joined to the film cradle [W] is a more complex construction as it is the interface between the main winding and shutter mechanisms (Figure 6-7). This element was also modified from the original model to allow external accessibility to the parameters of the gear disk (Shape1) that connects to the gear train.

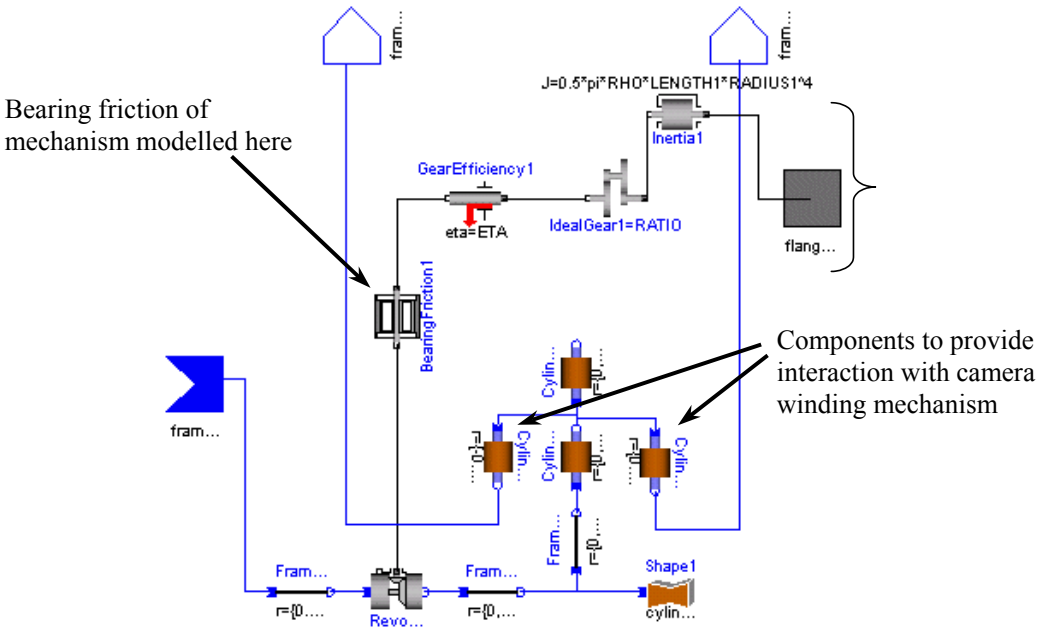

Figure 6-7: The end element [W] of the gear train that is connected to the film cradle
Components without annotation are explained in Figure 6-1; diagram adapted from (Bolognini 2003).

6.2.3. *Parallel grammar addition*

The winding mechanism described is conceptually similar to the clock design tasks considered in the chapter 5. The mechanism consists of axles (spindles in clock terminology) that are linked by spur gear pairs. The dimensions and power requirements of the objects produced are similar, therefore the scale of the design tasks can be said to be of the same order.

A major difference between the two systems is that there is no requirement on maintaining the gear ratio between gear disks while modifying the camera design. The

resultant behaviour after any changes is monitored by simulation. This is different from the clock example where the ratio constraints ensure that the clock hands behave in a manner that fulfils functional specification. Essentially a hard constraint has been relaxed and changed into a performance measure that can be used for design evaluation.

To enable ratio changing for a single gear pair, a new perturb rule is added to the existing library. In keeping with the previous assumption that there are a reasonable number of gear teeth evenly distributed about the gear disks, this new addition to the library is relatively straightforward, merely allowing the ratio of a gear pair to be changed by adjusting the pitch radii of the two disks. Information on the new perturb rule is summarised in Table 6-2 with a visualisation in Figure 6-8. Note that the distance between the two spindles remains unchanged as the rule is applied.

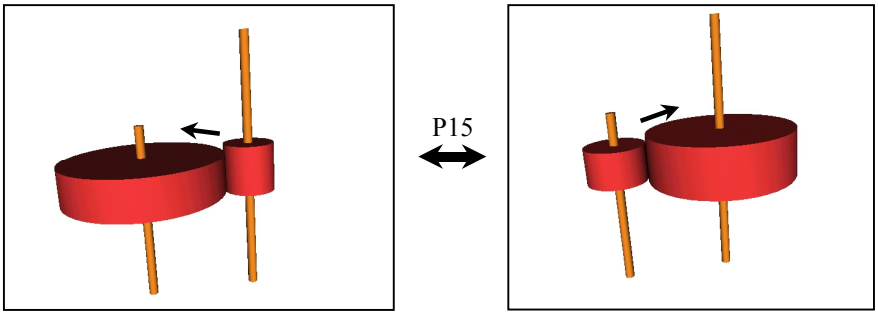

Figure 6-8: New addition to parallel grammar (perturb rule P15) for camera design

Table 6-2: Details of new perturb rule P15

P-Rule	Action	Rationale
P15: Change gear ratio	Changes the ratio of a gear pair. This alters the behaviour of the gear pair as well as changing its physical dimensions	Changing gear ratios alters the torque of the rotating elements. Enabling these values to be changed allows more preferential gear ratios to be sought out. Key constraint: mesh.

6.3. Performance feedback

As discussed in section 6.2.1, a Dymola simulation model consists of two parts, an executable and a parameter input file. The executable is pre-compiled using the

Dymola GUI for each design family that is to be considered. For the camera case study, nine separate simulation executables were prepared for a range of design families with two through ten components in the gear train. This was a straightforward process as an object-oriented model is used. All that was required was to edit the gear train class (Figure 6-6) to the required number of gear elements. If this had been a manual exercise, correct starting parameters for each of these additional new gear trains elements would have had to be determined. However, as these parameters will be generated by the parallel grammar this arduous process is not necessary.

Running Dymola simulations produces data sets for parameters in the model. As an example, Figure 6-9 shows the angle of rotation, θ, in the gear train of the three spindles furthest from the electric motor (Figure 6-6) in a photo-taking scenario. The shutter is exposed at about 0.9 s, after which the gear train winds the film on to ready the camera for the new shot. The different spindles rotate at different speeds due to the ratios in the gear train.

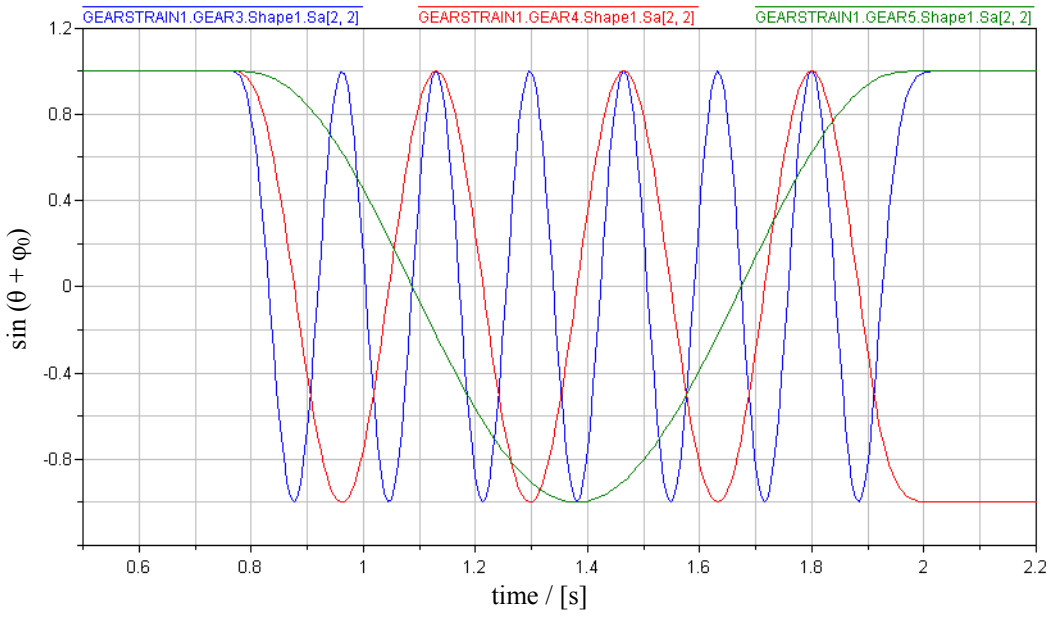

Figure 6-9: Angle of rotation θ of spindles in gear train during camera operation. φ_0 is a constant.

With Dymola simulation models in place, the only step still required is to output the desired performance data in a convenient format. An expedient method of analysing the data output from the simulation process is by studying the final parameter values at the end of the simulation run. This is an agreeable method for cumulative signals, for example the total energy drawn from a battery over a period of time. Current flow and charge usage simulation data for such an experiment is plotted in Figure 6-10. The final value can be read off a plot of function value against time to provide the required data, in this case just under 4 C of charge.

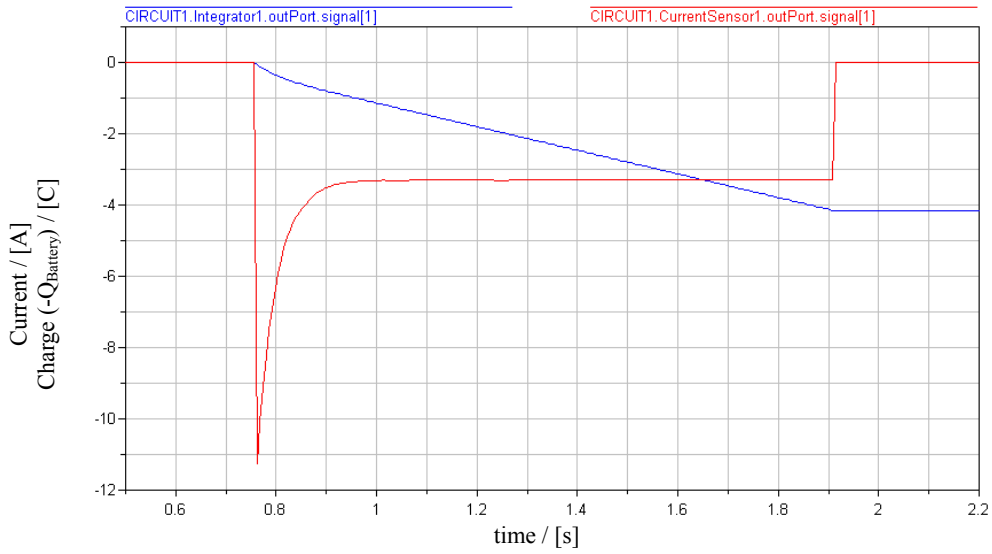

Figure 6-10: Power consumption from battery during camera winding operation.

However, other signals may not be obtained in such a simple manner. The blue trace in Figure 6-11 is a boolean plot showing when the electrical circuit is closed, indicating when current is being drawn from the battery to power the winding mechanism that rotates to prepare the camera for the next exposure. The exact moment at which this process completes is an important piece of information, as a short winding time means that the camera is readied for a new photo in less time.

As the data is plotted as a function of time, there is no mapping from function value to independent variable. In this case, a new parameter has to be added to the model that reacts to the particular event of interest and stores the time at which this occurs. In Figure 6-11, the signal t_{stop} ('time_stop' in diagram legend) does precisely this when the electric circuit reopens.

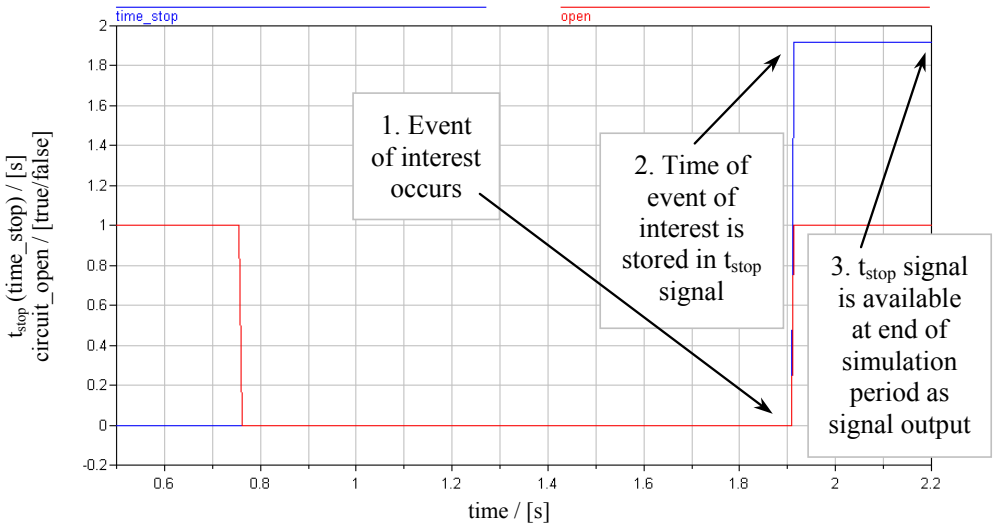

Figure 6-11: Performance-based evaluation – the t_{stop} signal output

For the camera design example, three performance variables, listed in Table 6-3, were considered. The t_{start} variable is not affected by changes to the winding mechanism, so the main performance-based metrics for this example are t_{stop} and $Q_{battery}$. In general it is desirable to reduce both of these values. Further implementation details on how the performance-based evaluation step is carried out are outlined at the end of the chapter (section 6.7).

Table 6-3: Performance-based evaluation variables

Variable name	Description	Rationale
PERF1 t_{start}	Time after simulation start at which the photo is taken.	An ideal camera takes a photo at the precise instant the user depresses the shutter release. Practically, a delay occurs at this point that is an undesirable performance characteristic.
PERF2 t_{stop}	Time after simulation start at which the camera has wound on the film and is readied for a new photo.	A quick winding time is beneficial to the user as it allows the camera to be used to take successive shots to capture a series of pictures.
PERF3 $Q_{battery}$	The total charge drawn from the battery during the winding process.	Long battery life is a desirable characteristic of products such as cameras.

6.4. A multi-objective hybrid pattern search

A side effect of using behavioural analysis is the possibility of a radical slow-down of the design generation process. Previously, evaluation data based on geometry metrics alone could be obtained simply and at no great computational cost. This justified the use of search algorithms whose advantages lie in their robustness and ease of implementation rather than rapid convergence characteristics.

Due to the combinatorial nature of searching for preferred designs, it is not feasible to use such wasteful algorithms if there is a significant computational cost association with obtaining evaluation data. It is therefore required to make better use of each design evaluation. Based on successful work on deterministic pattern search methods (Torczon 1997), such an approach was adapted for use with the parallel grammar to take advantage of improved convergence criteria over simulated annealing (Yin and Cagan 1998). A good body of work exists that has investigated the use of pattern search methods using rotation and translation moves in Cartesian space to tackle packing problems, from simple knapsack problems through to more complicated mechanical connectivity tasks such as clutch layout for automatic transmissions (Yin and Cagan 2000b; Yin et al. 1999). These algorithms employ heuristics, for example allowing component positions to be swapped, that have been found to be successful in making pattern search algorithms more efficient (Yin and Cagan 2000a).

6.4.1. Hybrid pattern search

An adapted method, referred to as a hybrid pattern search, is now presented for use with the parallel grammar. Traditional pattern search methods work by deterministically attempting ever-reducing steps in an attempt to improve the quality of the solution. Patterns of successful steps are repeated in attempt to accelerate the search progress towards an optimal solution. The Hooke and Jeeves (1961) algorithm is a commonly used pattern search method and an example of the basic algorithm is shown in Figure 6-12 for search in two-dimensional Cartesian space. Alternate increments and decrements in each variable are carried out with successful steps being repeated in combination as a pattern step. The steps are reduced in size if no

improvement is achieved until the step size is smaller than a pre-specified threshold value. This terminates the search.

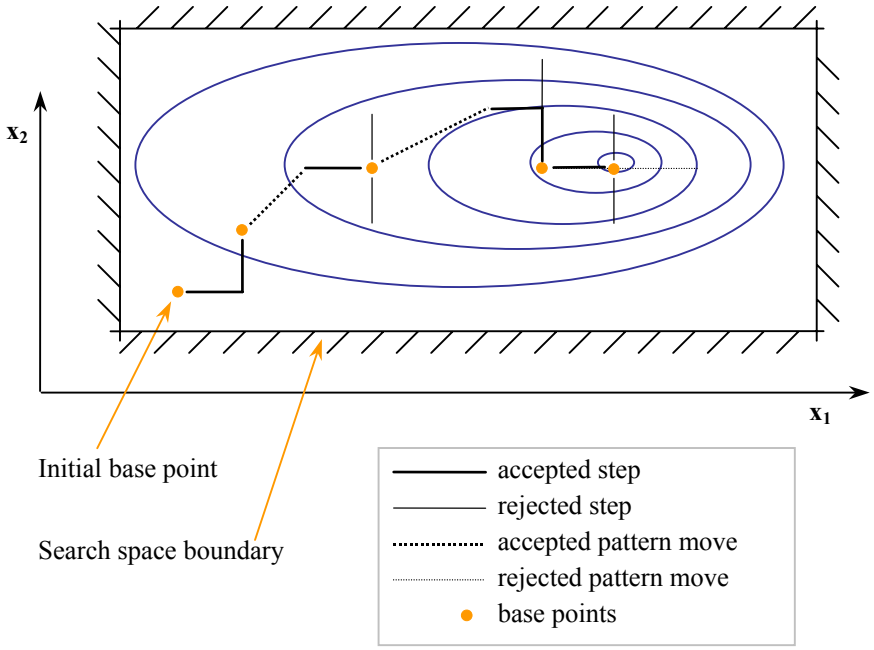

Figure 6-12: Hooke and Jeeves algorithm for a two-dimensional function space (Hooke and Jeeves 1961)

In the example in Figure 6-12 the successive base points show progress towards the best solution. The method can be generalised for search in n-dimensional space. To be applicable for design modification this search method must be adapted as, unlike the changing of co-ordinates in the example in Figure 6-12, it is not possible to modify the variables of a design freely as many changes would result in invalid designs. For example, changing the radii of connecting gear disks would, in almost all cases, result in a broken connection of that gear pair. Hence perturb rules, with their in-built constraint fulfilment, must be utilised as before to ensure validity of designs when changing design parameters.

A flowchart representation of the hybrid pattern search algorithm is shown in Figure 6-13. The general principle of the method is to find patterns of modification rules that are successful at improving designs as determined by performance evaluation. Once found, these patterns are attempted repeatedly to encourage rapid convergence to preferred designs. The algorithm has two main phases, (1) a local and (2) an extended pattern search phase. The local pattern search consists of choosing N_p perturb rules

(referred to as a sequence of length N_p) and applying these modifications in turn. After each successful change, the newly modified design is evaluated. Then, if the sequence is successful at finding an improved design, this pattern of rules is repeated with evaluation taking place only after the complete sequence of perturb rules has been completed.

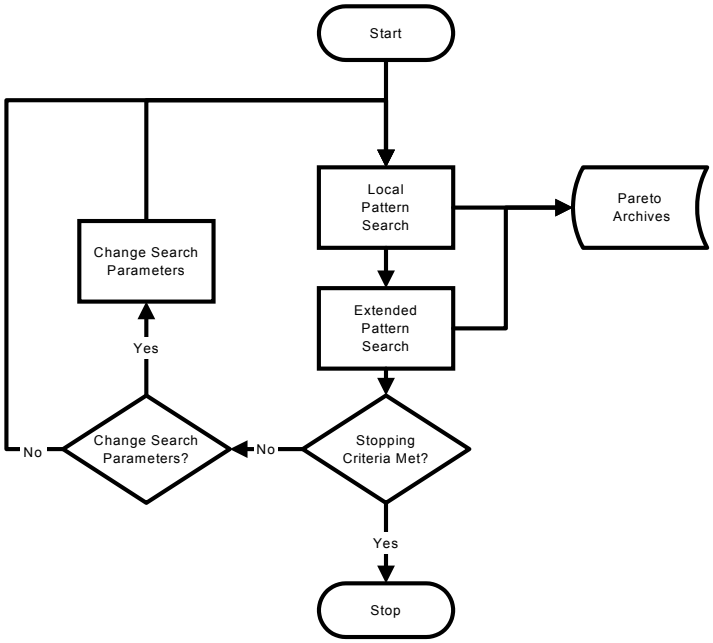

Figure 6-13: Flowchart of hybrid pattern search algorithm

The key variable for this search process is the parameter N_p, the discrete number of rule applications chosen to be applied as the pattern sequence. This is different from the step size $D_{i,x}$, a parameter that governs the range of perturb rule input values, for example the allowed increase in length of a spindle. As different perturb rules can be active on different scales, the subscript x in $D_{i,x}$ enables the specification of a step size D_i for different perturb rules x. The subscript i serves to differentiate the step size D_i from other parameters (see Table 6-7). Similarly to simulated annealing, decreasing refinements to designs are required and the step size $D_{i,x}$ is therefore reduced as the search algorithm progresses.

The extended pattern search is very similar to the local pattern search except that, during the first application of the pattern, evaluations are not carried out after every modification, but only after the full sequence has been applied. This introduces the

possibility of capturing sequences of modifications that initially result in worse performance only to enable subsequent modifications that result in a final increase in performance. This is shown schematically in Figure 6-14 (c.f. Figure 5-6). The sequence of three perturb rules result in an overall improvement in the objective function value, even though the first perturb rule modification increases the objective function value. This may increase the chance of escaping from local minima in design space (Vale and Shea 2003), as is found in simulated annealing search.

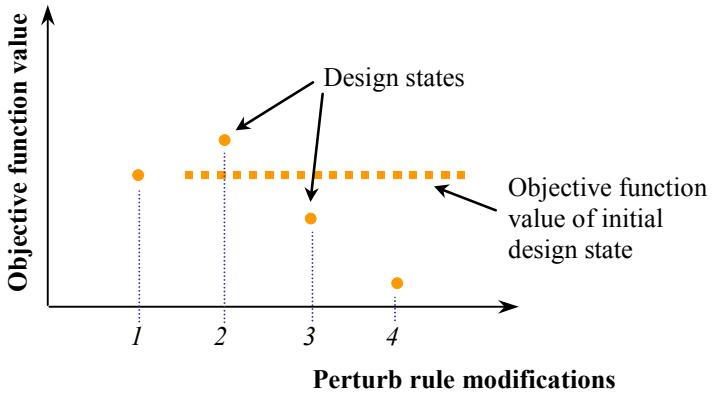

Figure 6-14: A non-monotonic sequence of objective function changes (schematic)

The hybrid pattern search is a non-deterministic search algorithm as the perturb rule parameters are not specified explicitly by a user but are generated at random within bounds determined by the pattern parameters. Parameter selection is outlined in more detail in the implementation part of this chapter where the algorithm is discussed in greater depth (section 6.7.2). Initial pattern step sizes are determined by the user and are reduced as design modification progresses.

6.4.2. Multi-objective search

The search algorithms used in chapter 5 sought out a single most preferred solution, using a metric or a combination of metrics to evaluate the quality of the design. This is adequate if the objective function evaluation matches user requirements.

However, design tasks are usually ill-defined problems. Consider the clock designs generated in the previous chapter (set 5C) for a combination of two different metrics. In this case, the metrics were complementary, i.e. both drove search roughly in the desired direction, but finding the perfect balance between the weightings of these two metrics in order to provide the best answer was not easy. It required the running of many separate experiments using trial and error to find the right combination that generated the best results. In this case, neither of the two objective functions used (the weighted and simple thickness metrics) perfectly matched the design goal. The use of objective functions in combination is not always intuitive and it is therefore difficult to predict what combinations are likely to prove successful.

In a complicated design task it may be a question of attempting to satisfice a large number of competing design objectives that are not all simultaneously attainable. Whether design objectives are complementary or are in conflict with each other may not even be possible to predict in advance. This is especially likely in situations where design synthesis is being used to explore completely new design spaces that a designer may not be familiar with.

In both cases, i.e. for complementary and competing objectives, using a true multi-objective method can be a useful method of finding good solutions to optimisation problems. For competing objectives, the aim is to produce an archive of designs that gives a trade-off between the design objectives under consideration, such as cost versus a behavioural performance aspect of some kind. This enables the maximum amount of information about the trade-off to be visualised to allow the designer to make an informed selection of a satisficing compromise. In the case of co-operating design objectives, the multi-objective approach is also advantageous as the method creates collections, termed archives, of preferred designs.

Design archives are the key to multi-objective search. Seeking out this information at the first time of asking provides more information at similar computational cost as doing one search using the previous single-objective methods. All designs created by the generation algorithm are submitted to a design archive to determine whether or not it is a ‘good’ design and should be kept. The ‘goodness’ of a design is determined by checking whether the new design is dominated by any other solution already in the

archive, i.e. a solution has not already been found that is better than the new solution with respect to all objectives under consideration. If the new design is not dominated by an existing design it is added to the archive. The archive is then re-examined to determine if the newly added design dominates any of the previously stored designs. Any such dominated designs are removed from the archive and discarded. Figure 6-15 shows the now complete synthesis algorithm diagram (c.f. Figure 4-13 and Figure 5-1) with usage of design archives for multi-objective evaluation.

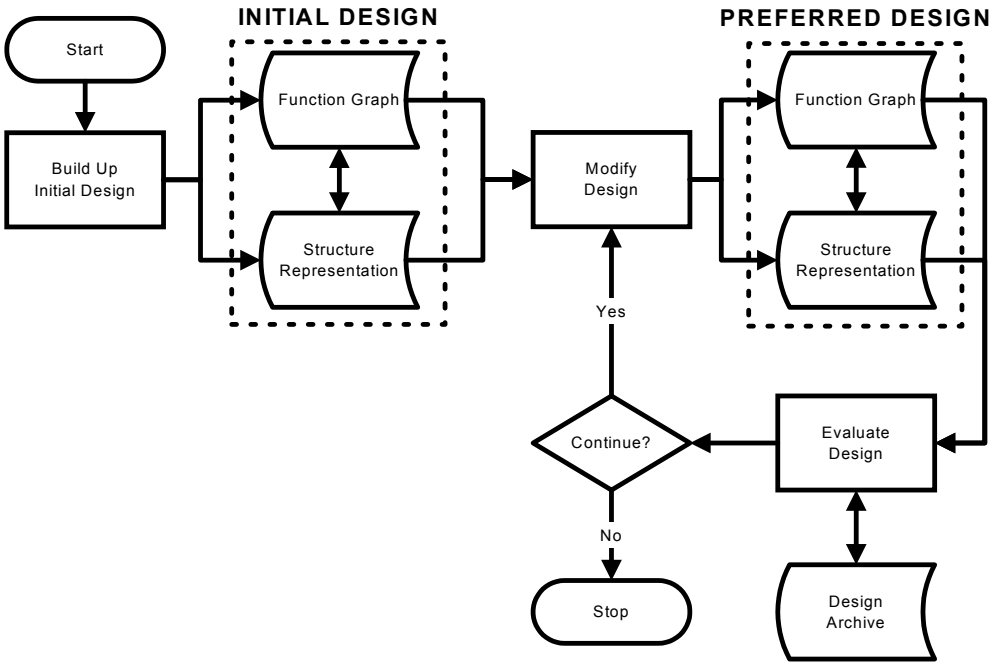

Figure 6-15: Multi-objective synthesis algorithm.

At any stage of search, inspection of the archive yields the best designs with respect to each objective function under consideration, as well as intermediate designs that show good trade-off characteristics between these objectives. This set of non-dominated solutions, named the Pareto[35] set, allows a large quantity of data to be presented in a convenient format. A great strength of this approach is that no *a priori* knowledge regarding the quantitative trade-off between the objectives is required for this method to be successful.

[35] The set of non-dominated solutions are named in memory of Vilfredo Pareto after his economics work on 'ophelimity'. This concept introduces the idea of an individual making a decision based on the desirability rather than the actual utility of the outcome of this decision. Maximum ophelimity describes a situation much like a Pareto front where the overall desirability of a system cannot be increased without a reduction elsewhere (Pareto 1906).

6.4.3. Hybrid search verification

The performance of the new hybrid pattern search method is investigated by reproducing the clock design experiment used for the main demonstration problem in chapter 5 (see section 5.3). Each run of the multi-objective algorithm creates an archive of best solutions obtained from the starting solution, so it is not necessary, as with the RD and SA algorithms, to carry out multiple separate experiments from each initial design. Where previously 50 search cycles generated 50 designs in total (5 separate initial solutions and 10 preferred design generated from each), only one search cycle is required to generate an archive from each initial design.

Design archives for 20 initial solutions are generated for the clock design problem in Figure 4-18. These 20 initial valid solutions are generated from scratch using the C-Rules of the parallel grammar. Search is then conducted from each of these to find preferred designs.

Despite producing more designs of interest this method requires less computational resources than the simple search methods used previously. The aim, as in section 5.4, is to generate thin designs. Three complementary objective functions are used, thickness, weighted thickness and compactness. To be accepted by an archive, a newly modified design must show an improvement in one of the evaluation functions over the designs already in the archive. Such non-dominated solutions are added to the archive and may, in turn, dominate existing designs in the archive. Therefore consideration of weighted thickness and compactness allows designs to be archived that do not necessarily show an improvement in thickness, the main objective, but whose changes with respect to the other design metrics are beneficial for future reductions in thickness.

The main user-controlled parameter of the hybrid pattern search is the length N_P of the pattern sequence (Table 6-7). This is the length of the chain of modifications that, when successfully completed, can be considered a pattern. A short pattern is more likely to be applied successfully and therefore may result in a large number of

successful modifications. A longer pattern, however, is more likely to find complex combinations of perturb rule applications that result in desired design changes. Hence different values of N_P were chosen for comparative purposes. For each of the 20 Pareto archives generated by the pattern search, the best metric values are chosen and tabulated in Table 6-4. The best thickness values are generated for the simplest pattern length (i.e. $N_P = 1$), with the average thickness values increasing with pattern length.

Table 6-4: Summary of hybrid pattern search results for thin clock designs (twenty design archives, 20,000 search cycles each)

	Pattern Length N_p			
Thickness [mm]	**1**	**2**	**3**	**4**
Average value	6.9	8.2	8.7	8.9
Minimum value	5.3	6.2	5.7	6.6
Maximum value	8.1	13.5	11.9	14.6
Standard deviation	0.8	1.6	1.5	2.3

The data from the design archives can be presented in many different formats. As three metrics have been used for this experiment, the Pareto set is effectively a two-dimensional surface in the space of the three evaluation metrics. Having generated the data summarised above, it is of interest to investigate the archives that contained the best results with respect to thickness, i.e. the minimum value in Table 6-4. The members of the best archives are plotted against their metric values in Figure 6-16. Metric values are normalised prior to search to be of the same order of magnitude.

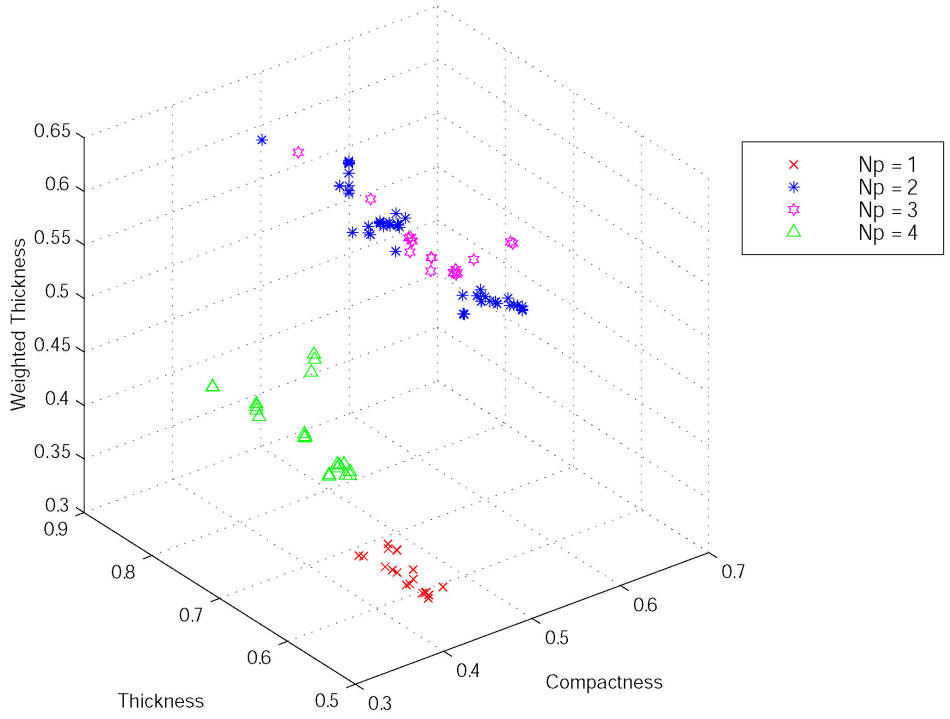

Figure 6-16: Best archives generated for different pattern lengths N_p

The best values in the archive are those that are nearest the front bottom corner, as this location corresponds to the minimum for all objectives. The two archives for $N_p = 2$ and $N_p = 3$ are intermingled, whereas the other two archives are more clearly divided in metric space. Visualising three-dimensional data in two dimensions is not easy , and Figure 6-16 is confusing in that the human eye attempts to project the plotted points onto the co-ordinate planes of the graph. This can partially be overcome using a dynamic viewer. This representation then becomes very effective, allowing a designer to investigate the location of particular points by moving the viewpoint of the three-dimensional around in space. For clarity, future data in this non-interactive thesis will be presented as two-dimensional projections.

Another interesting result from this experiment is that the best design produced by the hybrid pattern closely approaches the global optimum. From Table 6-4, the thinnest clock generated by the algorithm has thickness 5.3 mm, only a fraction more than the

absolute theoretical minimum calculated in chapter 5. This design is shown in Figure 6-17.

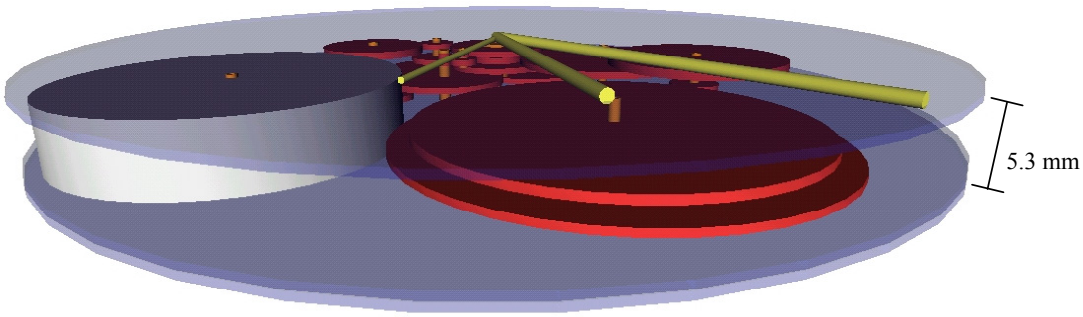

Figure 6-17: Very thin clock design (thickness 5.3 mm) – pattern step length $N_p = 1$, solution #5 from 19th design archive

This very thin design is at the limit of what is possible given the design constraints and is better than the previous best designs generated in section 5.4. The power source is the limiting factor with a minimum thickness of 4 mm. Each of the three support plates specified for this design has a thickness of 0.4 mm, so the design generated is the thinnest possible while allowing space vertically between the limiting elements. To obtain this thinnest design the algorithm has arranged all the gears and spindles between two of the support plates, sandwiching the middle plate to the bottom plate without any components in between. This is an interesting design, as it introduces the possibility of only using two support plates. Being able to peruse such design ideas as virtual prototypes is an example of the potential of the proposed synthesis approach to spark design innovation. Manual inspection of this model may allow the designer to consider new ideas such as removing the middle support plate and using the algorithm to generate favourable layouts that are stable with only the two remaining plates.

The potential merits of the multi-objective hybrid pattern search have been shown and verified through the previous example. The method generates better results for the test clock design problem using less computational resources. The generation of a design archive of Pareto solutions is useful as it provides a large amount of information about the design space in a concise manner.

6.5. Search results

Having compared the hybrid pattern search algorithm with previous simplistic methods (section 6.4.3) for verification purposes, performance-based evaluation was introduced to investigate the camera redesign problem introduced in section 6.2.2.

6.5.1. Performance-based results

Pareto sets of possible camera designs were generated using a variety of different combinations of objective functions. The perturb rules used previously were employed as well as the new modification rule P15 for the camera domain (Figure 6-8). Perturb rules P12 and P13, the rules that allow insertion and deletion of new spindles into the design, allowed component topology to be altered and therefore enabled different design families to be explored by this search method. This is of particular relevance to the camera redesign problem as it would be interesting to be able to gain insight into the design rationale for designing the camera gear train with five separate spindles (Figure 6-6). Such solutions are referred to as '5-noded' solutions. Why were '4-noded', '6-noded' or other design families not chosen for the original design? As further verification of the method it is desired to generate such significantly different designs, i.e. from different families, to explore their performance merits with respect to the original design.

Rather than just plot the pure Pareto front of generated designs, the non-dominated solutions for each individual design family are stored. This is done by using a Pareto archive as well as a so-called 'noded Pareto archive'. Each new design is submitted to a family-specific 'mini-Pareto archive' as well as the overall Pareto archive. For example, a newly generated 7-noded solution is submitted to the overall Pareto archive to see if it a non-dominated solution compared to all other designs. It is then also submitted to the noded Pareto archive consisting only of 7-noded solutions, to see whether it is dominated by all previously found 7-noded solutions. Acceptance by a noded Pareto archive is used as a criterion for determining a successful design modification. This method allows the algorithm to explore the limits of each design family as well as merely looking for the overall best-performing designs.

Figure 6-18 shows all the solutions in the noded Pareto archive for the two main performance-based design objectives, $Q_{battery}$, the charge removed from the camera battery to wind on the device after taking a photo, and t_{stop}, the time at which this process is completed. It was expected that these two performance metrics should be competing objectives and this is borne out in the diagram. The 4-noded solutions are effectively the overall Pareto front, but including the noded Pareto sets adds more information about the space of alternatives, showing the trade-off between $Q_{battery}$ and t_{stop} for different design families. The relationship between these two objectives is almost directly proportional as other component parameters, such as mass of elements, have only a weak effect on these characteristics. This relationship was determined by the search algorithm without *a priori* knowledge about the detailed mechanism controlling this relationship.

The original design has t_{stop} values of 1.9 s (Bolognini 2003). From Figure 6-18 it can be seen that this presents a good trade-off between the two objective functions for 5-noded solutions. Increasing the number of gear elements increases the potential losses from each component, demonstrating that a lower number of nodes is a more preferred design strategy. Hence, it might have been expected that a 4-noded solution would have been used, as the plot shows that a design with only four gear elements in the gear train shows better design characteristics. To answer this question further investigations must be carried out.

Figure 6-19 shows multi-objective design archives for t_{stop} and mass objectives, as well as the initial solutions that served as the starting points for the design archive. These starting solutions are the initial designs generated using C-Rules after which design modification takes place using perturb rules, or P-Rules (see section 6.4.3). Note how these initial solutions all have the same value of t_{stop} as the initial design generation procedure used ratio information from the original design. This plot helps explain why a 4-noded solution may not have been used for the original camera. Due to the larger gear disk size required to bridge the constant gap between the ends of the gear train if only four gears are used, the 4-noded Pareto has greater mass than other solutions, such as the 5- and 6-noded data sets. Hence a 5-noded solution might have been chosen to avoid using gear disks with overly large radii. Using a gear train with five elements therefore presents a good trade-off between mass and t_{stop}. A plot from a

different experiment for the same objective functions shows similar results, albeit with lower mass solutions near the left of the diagram (Figure 6-20).

Figure 6-21 shows data for aspect ratio and battery usage. The aspect ratio metric is of particular relevance as the camera gears are made of plastic and are therefore likely to be injection moulded (Ulrich and Eppinger 1995), in which case a low aspect ratio of parts is desirable. The 4-noded solutions are to the right end of the plot, indicating again the increased radius of the gear disks for these solutions. The results are less clear-cut than for data sets shown previously. Note the lower $Q_{battery}$ values of some of the 5- and 6-noded solutions. Figure 6-18 shows that cameras with gear trains designed to these specifications would have long winding times, e.g. in excess of 5 s.

Not all data produced by the multi-objective method provides as much information. Figure 6-22, a plot for battery usage against mass, shows no real trade-off between these two variables. The vertical line of design solutions shows there is no great dependency between mass and the number of nodes in the gear train. The best solution by far is the 4-noded solution in the bottom left of the graph, which might explain the poor performance of high-noded solutions, e.g. the 9- and 10-noded solutions in the top right hand corner.

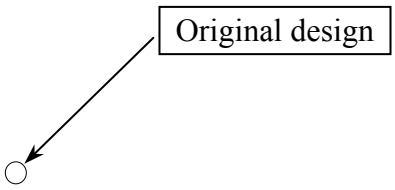

Figure 6-18: Multi-objective camera search results for t_{stop} and $q_{battery}$ (data set 01450)

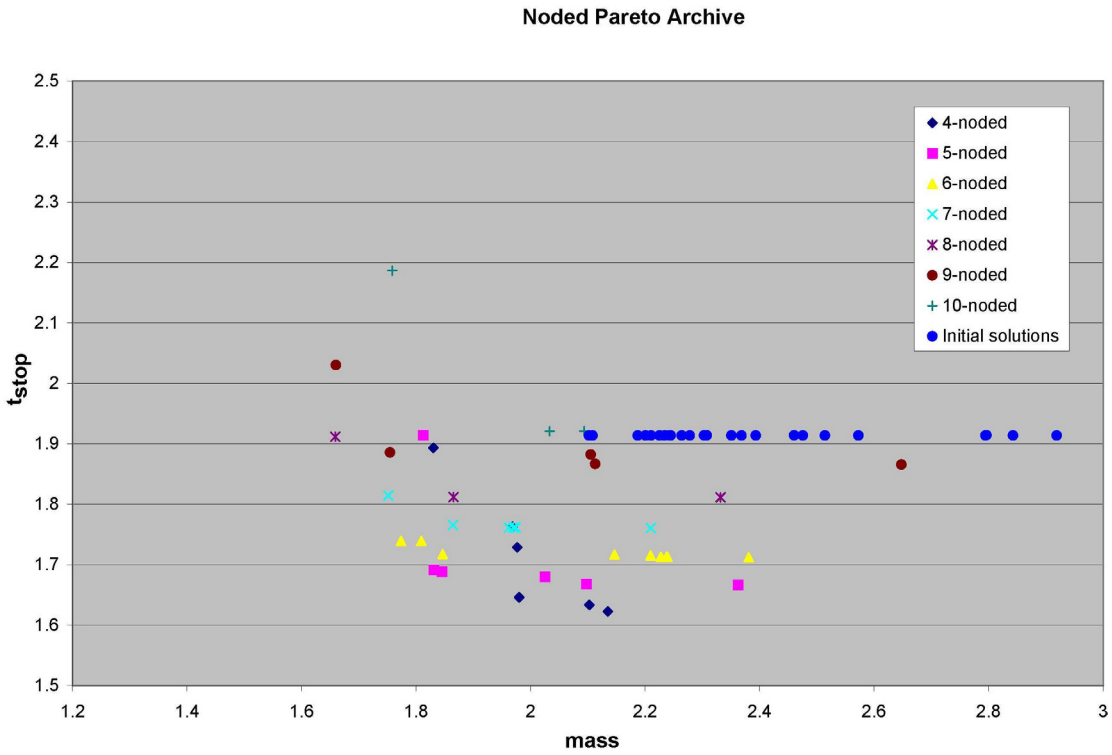

Figure 6-19: Multi-objective camera search results showing initial solutions (data set 16042)

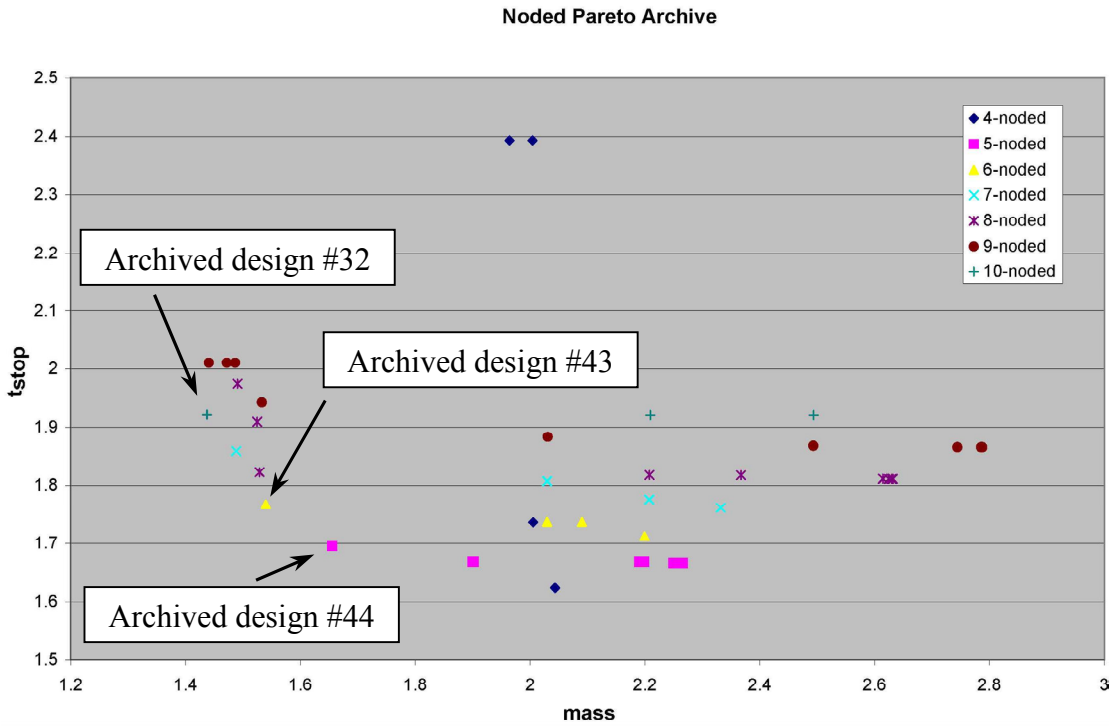

Figure 6-20: Multi-objective camera search results for mass [scaled metric value] and t_{stop} [s]

(data set 16092)

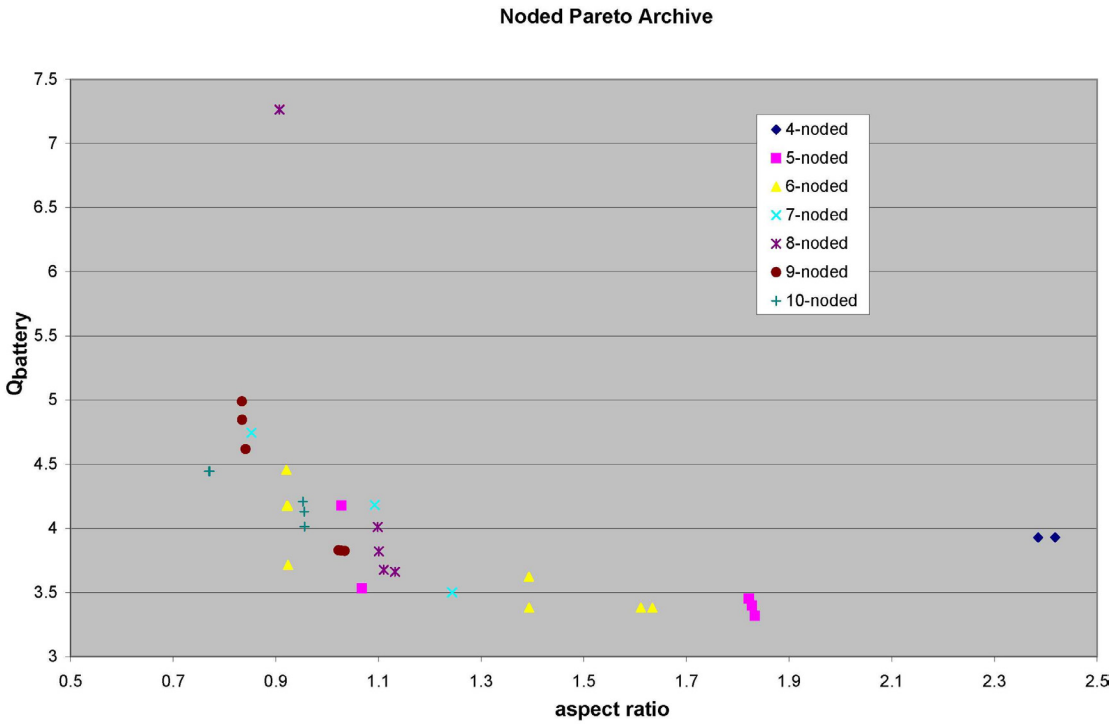

Figure 6-21: Multi-objective camera search results for aspect ratio and battery usage (data set 30691)

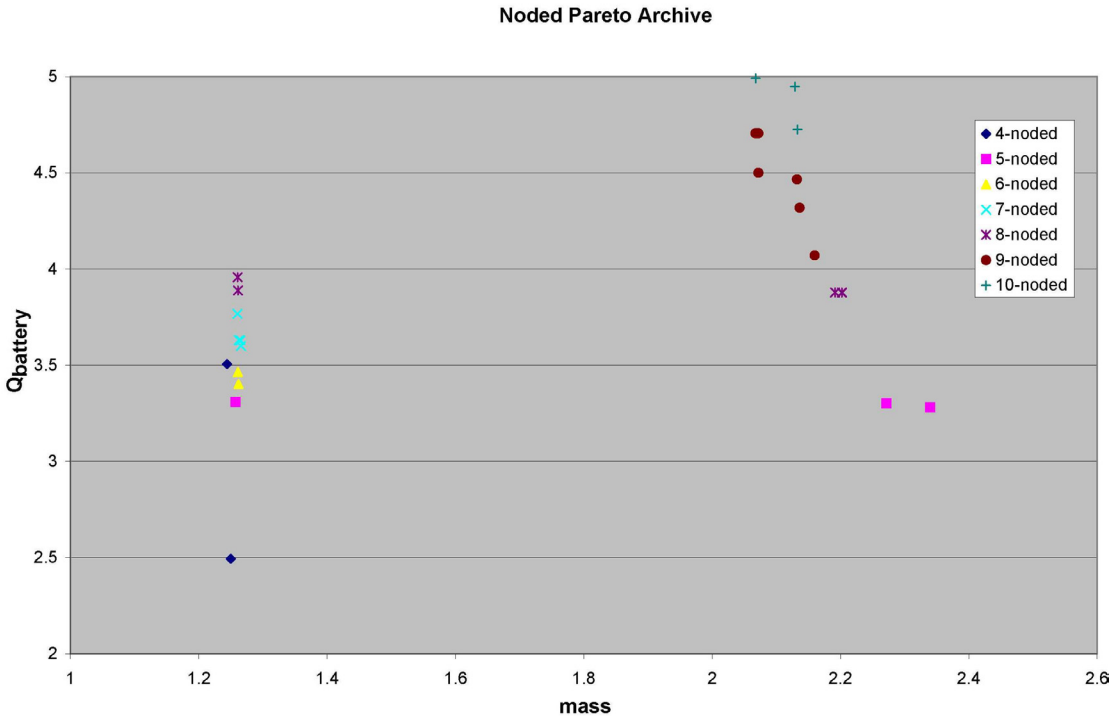

Figure 6-22: Multi-objective camera search results for mass and battery usage (data set 30698)

Two Pareto designs from Figure 6-20 are shown in Figure 6-23. The light blue boxes and cylinders in these pictures show existing hard boundary constraints such as the existing camera housing, the film and the winding motor. The pictures are snapshots of the virtual prototypes generated automatically for each solution in the archive.

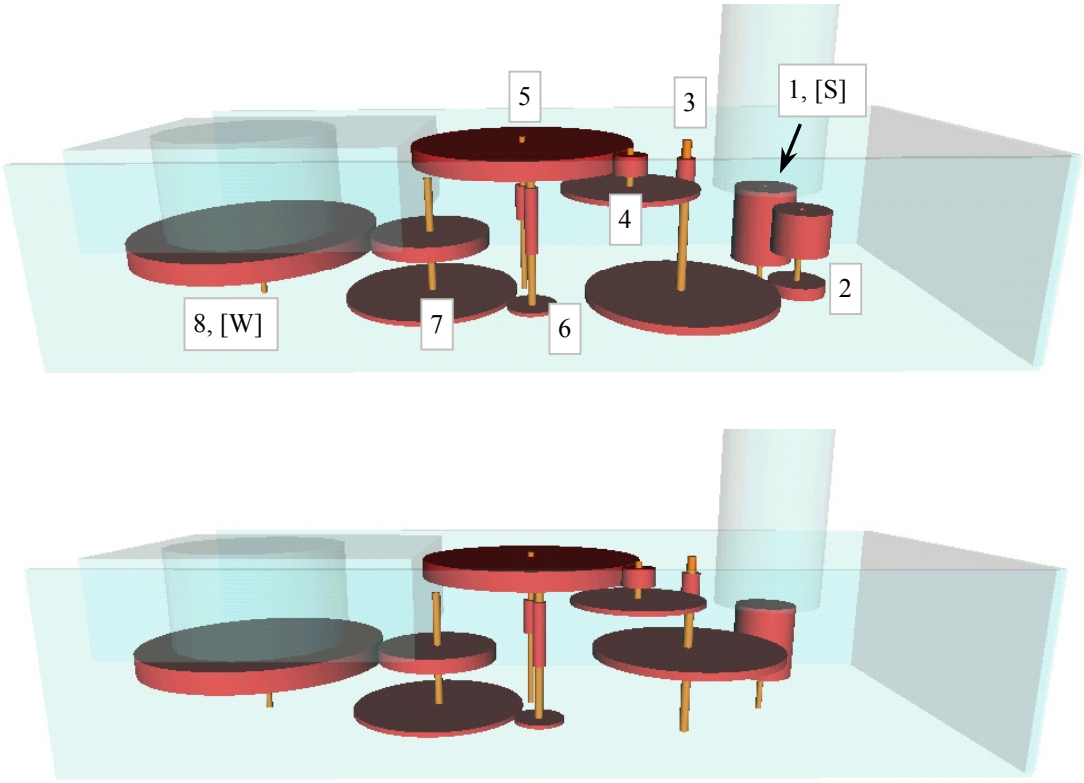

Figure 6-23: Pareto solutions #43 (top) and #44 (bottom) from data set 16092 (c.f. Figure 6-20)

Design #43 is a 6-noded solution; design #44 is a 5-noded solution. The former has been labelled to give an overview of what the components are; the latter has been left unlabelled to allow a clearer view of the structure. The proximity of the designs in the Pareto archive is not surprising as they are quite similar: one of the designs was created from the other. Spindle '2' in design #43 does not exist in design #44, where spindles 1 and 3 are connected directly. The remaining components are the same.

Figure 6-24 is another design generated in the same experiment. It is a 10-noded solution, resulting in a configuration with many parts. Due to its high part count, it is

unlikely that such a design would be considered for further development for this particular layout of boundary constraints. However, if low-noded solutions were not so easy to design due to a more constrained spatial layout, being able to generate high-noded solutions such as that in Figure 6-24 could be of benefit.

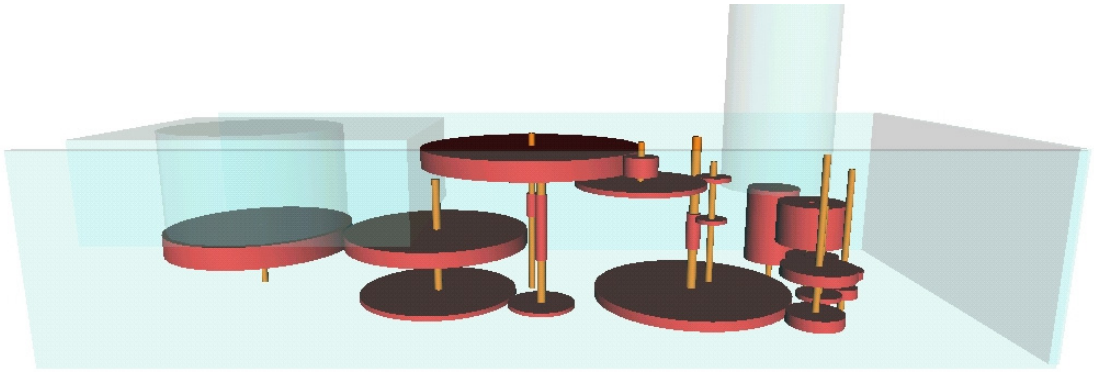

Figure 6-24: Pareto solution #32 from data set 16092
(c.f. Figure 6-20)

6.6. *Conclusions*

This chapter has addressed some additional requirements of performance-based mechanical synthesis. The main objective was to enhance design evaluation to judge the quality of designs created by the parallel grammar in terms of behaviour criteria as well as the previously-used geometric metrics (chapter 5). The use of computational behavioural modelling achieved this aim, enabling actual performance parameters to be considered by the evaluation step of the design synthesis framework.

A new case study, a camera gear train redesign task, was introduced as a test case for behavioural performance-based analysis. An existing camera design was analysed, alternative design configurations were considered and virtual prototypes of these were generated. Evaluation hinged on simulation of parametric designs generated by the parallel grammar. In combination with existing geometry-based metrics a series of designs were generated to understand the rationale for the original camera. Data from this process provided information on the performance envelope of the camera and enabled new design possibilities to be considered.

To take full advantage of the performance data from the simulation environment a hybrid pattern search method was introduced that performed better than the random downhill and simulated annealing search methods used previously (c.f. sections 5.4 and 6.4.3). This new search algorithm is well-suited to finding good designs through the modification of constrained design problems. The algorithm is tailored for use with a perturb rule library introduced in chapter 5.

It has still not proved possible to generate new clans of designs computationally. The current implementation requires a user to choose the system topology of a design that the parallel grammar takes as functional specification to generate initial designs. As useful as it is to be able to generate new layouts for design families and even mediate between different families during search, it would be a truly exciting prospect to be able to abstract the specification one level higher to allow computational generation of new design clans that can then be taken forward to virtual prototypes.

This chapter can be considered as a proof-of-concept demonstration for behavioural simulation-based feedback to drive computational design synthesis for the purposes of creating designs with desirable characteristics. The next question is to consider whether this work can be applied beyond the simple clock and camera case studies. This is the topic of chapter 7. Vehicle gearbox design is considered as a design field that is likely to see increasing use of computational synthesis methods over the next few years.

6.7. Implementation details

In line with the computer aided engineering design methodology discussed in chapter 2 (Figure 2-2), scripting languages were used to provide the link between previously incompatible software (Ousterhout 1998). The implementation of the performance feedback code made extensive use of Perl and Bash scripts to mediate between the original C++ generation code and the Modelica analysis software. This approach has resulted in a flexible toolbox for accessing and running simulations in the Dymola modelling environment that could also be used with other design generation code.

6.7.1. Behavioural performance feedback

The implementation of the performance-based design method hinges on code that is used to pass information between the generation and analysis software packages. The files, variables and C++ functions used for this are listed in Table 6-5 and Table 6-6. The process can be split into three tasks, (1) exporting the new design from the generation software, (2) analysing this design and (3) re-importing the performance evaluation data into the generation software. These steps are controlled by the three C++ functions listed in Table 6-5 and are outlined in detail in Figure 6-26.

Table 6-5: C++ functions and run-time variables for performance feedback

C++ Function	Run-time variables
`make_changes_file`	`$1` – machine to use for simulation
`get_performance_data`	`$2` – number of nodes in solution
`read_performance_data`	`$3` – Process ID (PID)

As discussed above, file scripts (written in Perl) were used for flexible manipulation of text files to quickly strip data from files and replace new parameter values in an efficient manner. The use of scripting languages enabled a relatively seamless interaction between design generation code (Linux-based) and analysis software (Windows-based). Simulations were run separately from the generation platform on a dedicated analysis machine (AMD Athlon 2600+); bash scripts were used to transfer files between machines. Time lost through file transfer was offset by a faster simulation time due to this distribution of synthesis and analysis tasks to separate machines.

Figure 6-25 shows a screenshot of the camera model in the animation window of the Dymola modelling environment. The diagram is annotated to highlight key parts of the model. The 6-noded model shown is adapted from the original 5-noded model developed by Bolognini (2003).

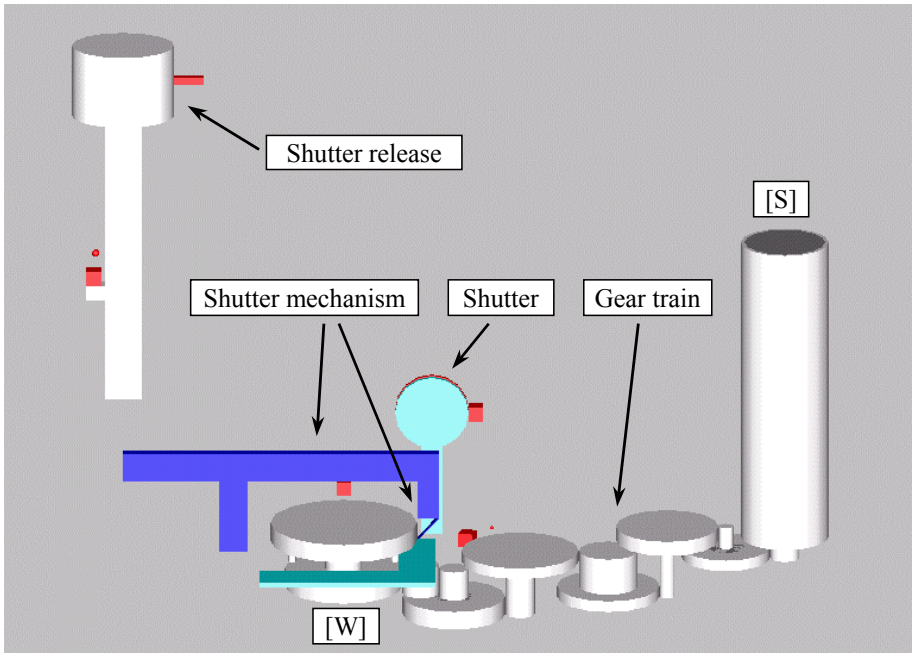

Figure 6-25: Modelica model of Vivitar CV50 as animated in Dymola modelling environment, adapted from (Bolognini 2003).

Table 6-6: Files used for performance feedback implementation

File	Type	Detail
`watchdog.pl`	Perl	Controls failed ssh executions when running simulations on separate machines
`change.pl`	Perl	Uploads new design parameters to Dymola input file
`get_results.pl`	Perl	Reads required results from Dymola results file
`send_node_A`	Bash script	Scripts used for file transfer when simulations are being run on dedicated analysis machine
`send_node_B`	Bash script	
`send_node_C`	Bash script	
`send_node_D`	Bash script	
`changes_000PID`	Text	Exported data from generation code with information about parameter changes
`wd_status _000PID.txt`	Text	Control file that holds information about success or failure of ssh processes
`diagnostic`	Text	Performance variables requested by user
`perf_data_PID.txt`	Text	Actual data values for performance variables
`dsin.txt`	Text	Dymola simulation parameter input file
`dsfinal.txt`	Text	Dymola simulation parameter file of final results
`dsres.mat`	Matlab	Dymola simulation results
`dymosim_$2.exe`	Executable	Dymola simulation executable

- **make_changes_file**
  - Takes current code-generated solution and writes geometric parameters to **changes_000PID**
- **get_performance_data**
  - Calls **send_node_A $1 $2 $3**
    - Retrieves particular Modelica **dsin.txt** input file for **$2** nodes to **xdsinPID.txt**
    - Runs **change.pl $3** to vary parameters in **xdsinPID.txt** as specified in **changes_000PID**
    - SSH copies **xdsinPID.txt** to simulation machine as **dsin.txt**
  - Calls **send_node_B $1 $2 $3**
    - SSH runs Modelica simulation file **dymosim_PID.exe** on machine **$1**
  - Calls **send_node_C $1 $2 $3**
    - SSH copies Modelica final results file **dsfinal.txt** to local file **dsfinalPID.txt**
  - [option] Calls **send_node_D $1 $2 $3**
    - SSH copies Modelica data file **dsres.mat** to local file **dsresPID.txt**
  - Runs **get_results.pl**
    - Reads required performance variables from **diagnostic**
    - Strips these data values from **dsfinalPID.txt** and writes them to **perf_data_PID.txt**
- **read_performance_data**
  - Returns performance data from **perf_data_PID.txt** as scalar function

Figure 6-26: Performance feedback file structure

6.7.2. Hybrid pattern search details

The hybrid pattern search algorithm takes a variety of parameters that can be set by the user to fine-tune the search for improved results. These are detailed in Table 6-7. The algorithm is outlined in C++ pseudocode code in Figure 6-27.

Table 6-7: User-defined parameters for hybrid pattern search algorithm

Variable	Explanation	C++ variable name	Example value
N_s	Number of initial starting solutions generated using the parallel grammar	SOLUTIONS	15
N_e	Number of final preferred archives generated from each initial solution	EXPERIMENTS	1
N_C	Maximum number of search cycles permitted for each experiment	SEARCH_CYCLES	100
N_P	Length of pattern sequence	NNN	2
N_{app}	Number of perturb rule applications permitted on each search cycle	N_APP	100
N_{base}	Number of cycles without successful changes before picking a solution from the archive and continuing from the search from this design	RETURN_TO _ARCHIVE	50
N_{param}	Number of cycles without successful changes before reducing the pattern step sizes $D_{i,x}$	COOL_PARAMETERS	5
D_{step}	Factor by which pattern parameters are reduced	COOL_RATE	0.8
$D_{i,x}$	Pattern step sizes, where x is the group of the perturb rule	X_STEP	2 – 8
$D_{floor,\ x}$	Pattern step lower limit	X_MIN	0.1 – 1

The following text is pseudocode for the hybrid pattern search algorithm.

```
// Start of hybrid pattern search pseudocode

// Declare parameter flags
// Flag used to check if sequence is being applied successfully
bool pattern_application = true

// Initialise archives
Initialise noded Pareto archives A_p,i
Initialise Pareto archive A_p

// This is the current design to search from
design = current design

while (stopping criteria are not met)
```

```
// Local Pattern Search
pattern_application = true
for (int i=0; i<N_C; i++)
        Select a sequence of length N_P of Perturb Rule types at random
                for (each of these Perturb Rules in the sequence)
                        Apply this Perturb Rule to design using range from D_i,x and
                                                at a random application point
                        while (application of this perturb rule is unsuccessful)
                                Apply this Perturb Rule to design using range from D_i,x
                                        and at a random application point
                                if (no of attempts at completing while loop > N_app)
                                        pattern_application = false
                                        break outer while loop
                                end if
                        end while
                        if (pattern_application == true, i.e. design changed
                                                                successfully)
                                Submit changed design to archives A_p,i and A_p
                                if (submission of design to A_p is unsuccessful)
                                        break for loop
                                end if
                        end if
                end for

                if (sequence of Perturb Rules was successful)
                        pattern_application = true
                        while (pattern_application == true)
                                Repeat this sequence of Perturb Rules to design
                                while (application of this sequence is unsuccessful)
                                        if (attempts at completing while loop > N_app)
                                                pattern_application = false
                                                break outer while loop
                                        end if
                                end while
                                if (design changed successfully)
                                        Submit changed design to archives A_p,i and A_p
                                        if (submission to A_p is unsuccessful)
                                                pattern_application = false
                                        end if
                                end if
                                if (number of applications > N_app)
                                        break while loop
                                end if
                        end while
                end if
        end for

        // Extended pattern search
        for (int i=0; i<N_C; i++)
                pattern_application = true
                while (pattern_application == true)
                        Select a sequence of Perturb Rule types (random)
                        Partner this sequence with application points (selected at
                                                                        random)
                        Partner this sequence with application range from D_i,x
                        // Apply this sequence of Perturb Rules to design
                        while (application of this perturb rule sequence is
                                                        unsuccessful)
                                if (no of attempts at completing while loop > N_app)
                                        pattern_application = false
                                        break outer while loop
                                end if
                        end while
                        if (design changed successfully)
                                Submit changed design to archives A_p,i and A_p
                                if (submission to A_p is unsuccessful)
                                        pattern_application = false
                                end if
                        end if
                        if (number of loops on while > N_app)
                                break while loop
                        end if
                end while
        end for

        if (no design successfully submitted to A_p in above)
```

```
                nochange_counter++
        else
                nochange_counter = 0
        end if

        // Return to base
        if (nochange_counter > N_base)
                Randomly select a design D_a from A_p
                design = D_a
        end if

        // Pattern parameter reduction
        if (nochange_counter > N_param)
                Reduce pattern parameters by specified step length
        end if

end while

// End of hybrid pattern search pseudocode
```

Figure 6-27: Pseudocode for hybrid pattern search algorithm

7. Industrial applicability

This chapter investigates the industrial applicability of the synthesis work in this thesis and, specifically, the parallel grammar as a means of design generation. Previous chapters built up a parallel grammar formalism for mechanical design synthesis. The parallel grammar was implemented computationally to investigate simple synthesis tasks, each chosen to address key issues in computational mechanical synthesis. The proposed formalism is now taken a step closer to practical tasks by studying two industrial design examples, with specific reference to the storyboard (section 1.2.1).

The parallel grammar is now expanded to represent the language of spur gear systems, which introduces the concept of clutches, i.e. allowing axles to be connected by more than one gear pair. Clutches allow shifting between different gear trains to provide alternative overall gear ratios. A multi-speed power drill is demonstrated as an initial example followed by the generation of a novel gearbox design for a vehicle transmission system. The parallel mechanical grammar is verified through the recreation of transmission layouts for current generation vehicles. Validation of the grammatical approach to synthesis is provided by assessing how it addresses the needs of a leading automotive transmission design company.

7.1. Validation of the parallel grammar

The clock and camera design case studies explore several challenges of computational synthesis applied to mechanical design. The parallel mechanical grammar was shown to be successful at recreating existing designs as well as enabling the discovery of preferred new designs.

Moving on from the exploratory case studies it is now desired to place the method into the context of current synthesis tasks that are relevant to industry today. Where the previous case studies provided verification of the parallel grammar and its place in the computational synthesis framework, new case studies with greater industrial applicability now validate the approach and demonstrate the potential utility of design synthesis tools.

Consider the parametric synthesis framework introduced in chapter 2 (Figure 2-3), the four phases of which have been addressed in previous chapters. After initial investigation of design case studies, the parallel grammar was used to generate both existing and new designs (chapter 4). Evaluation was considered on different levels, firstly using a geometric performance metrics alone (chapter 5) and then secondly using simulation-based, behavioural performance (chapter 6). In these chapters, stochastic search was used to mediate between design configurations within the described language.

This chapter takes the thesis full circle with the investigation of new design problems for validation purposes. Two case studies are considered, the design of a corded power drill and the design of vehicular power transmissions, specifically the generation of possible layouts for 5-speed transaxle gearboxes.

7.1.1. Power drill design

The generic term ‘drill’ refers to a tool that makes cylindrical cavities in bulk solids. Drills range in size from small devices for creating holes in wood or masonry through to large-scale machines for heavy industry that can be used to drill for oil or excavate tunnels for rail and road transport links. For the purposes of this case study the former end of the spectrum will be considered, in particular drills that are often referred to as ‘power drills’, i.e. hand-held electrically-powered drills that are operated by a single user. These machines are popular both with professionals, such as builders and carpenters, and amateurs, such as hobby enthusiasts. Power drills exist in many guises, for example as corded drills that rely on a supply of alternating current (AC) to power an electric motor. Battery-powered, i.e. direct current (DC), tools are also

common: these allow greater mobility while in use. Input power produces rotational motion that is transferred to a drill 'chuck' that securely holds a drill 'bit', the part of the tool that makes contact with the solid being drilled. Drill bits are often short-lived as they either snap or are worn out by abrasion, necessitating a chuck that enables quick replacement of worn drill bits as well as ensuring safe operation.

A particular drill was chosen for this case study in conjunction with design engineers at Black and Decker, a global power tool manufacturer. The KR850CRE (Figure 7-1) features a power output of 850 watts at a voltage of 230-240V, a user-controlled choice of output rotation (clockwise or anti-clockwise), optional hammer-drill mode and a two-speed gearbox. First gear provides high torque but a lower maximum rotational velocity, while second gear provides high maximum rotational velocity but at a lower torque. The gear selector can be seen in Figure 7-1, where the two gears are labelled '1' and '2'. In hammer-drill mode, the device leaks some rotational kinetic energy to effect a high-frequency axial motion to 'hammer' down on the point of impact with the bulk solid. This is a very effective way of drilling masonry as the compressive stresses introduced by this action cause the bulk solid to crumble. Drill bits with hardened tips are usually required for this mode of operation.

Figure 7-1: Black and Decker KR850CRE corded drill

The KR850CRE corded drill was reverse engineered for comparison with an existing exploded component diagram (Figure 7-2).

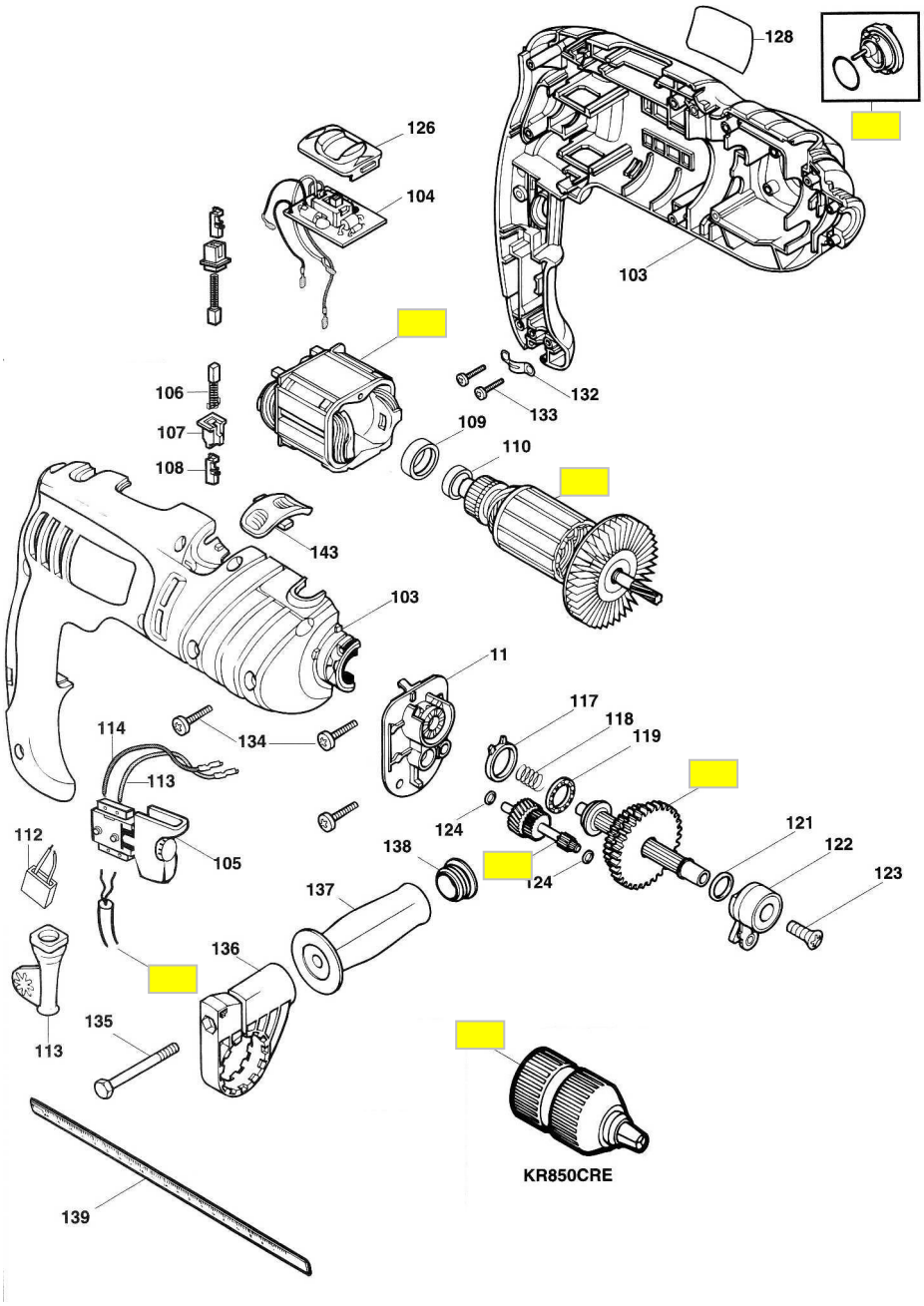

Figure 7-2: Exploded part diagram of corded drill[36]
Parts mentioned in text are highlighted.

[36] From product user manual. Also available at http://www.2helpU.com (last accessed 30 November 2003)

The architecture of the product is straightforward. Power (#112 – numbers refer to Figure 7-2) enters through the handle and supplies the electric motor (#101, #102) that sits atop the handle. This is the main mass-contributing component in the design. Apart from the rotating shaft of the motor there are two further rotating axles (#120, #125), connected by gear pairs, that lie inside the drill with the chuck (#140) attached at the far end of the drill body. The ratio between the second and third rotating axles can be altered by activating the clutch (#127) that slides the gear elements on shaft #120 to engage one of the two possible gear pair combinations.

From a mechanical design perspective the aspect of interest in this case study is the multiple gear train controlling the high/low torque output of the drill (Figure 7-3). The questions that could be addressed with a computational synthesis approach are whether alternative layout options or a change in the existing gear ratios could maximise operating performance. One can envisage further developments allowing detailed modelling of a drill to allow wholescale redesign options to be considered, such as studying the performance of the selected motor. However, at this point it is desired to place the case study in the context of practical industrial development where the designers use off-the-shelf components rather than consider parametric design variations.

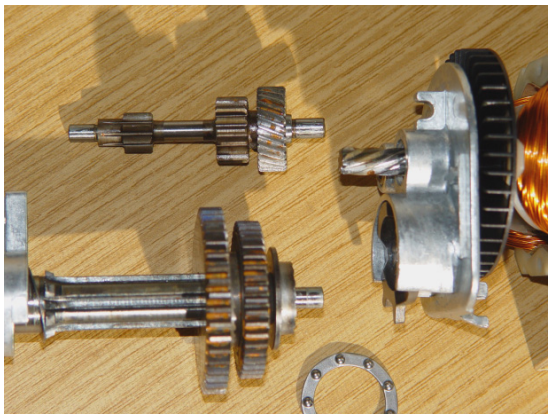

Figure 7-3: Close-up detail of exploded drill gear mechanism

At Black and Decker, the design of new products, e.g. corded drills such as the KR850CRE, is driven by two main design teams. An ‘outer’ team works on the design of clam-shell bodies with the dual functions of providing a stable base for components parts and providing the user interface for the product. Rapid prototyping

is used to create selective laser sintering (SLS) models of the outer form that can be produced to a level of detail similar to the final product. The 'inner' team is responsible for selection and placement of electromechanical components to fulfil the functional design specification. In practice, the 'outer' team are always looking to shrink the design envelope while the 'inner' team demands more space for the components of the product, resulting in a two-way 'tug-of-war' to find the best satisficing compromise (Hewitt 2003).

7.1.2. Vehicle gearbox design

Internal combustion engines used in vehicles have narrow operating ranges where torque and power are at optimal levels. Therefore to provide a vehicle, such as a car, with a useful range of speeds, a gearbox is required. A gearbox contains a number of parallel gear trains of differing ratio that can be selected, one at a time, to transfer power from engine to driven wheels to suit the driving conditions. The storyboard in the introduction considers the design of a transmission system for a racing car as inspiration for this research. The design of an automotive manual gearbox is now used as a final industrial case study.

Figure 7-4 shows a simplified sketch of the forward portion of a passenger car. Front wheel drive is commonly used for non-speciality motor vehicles, which in most cases means the engine is located at the front. Vehicle dynamics dictate that the heavy engine block is centrally mounted, resulting in restricted space on both sides of the engine. Sitting on one side of the engine in one of these restricted spaces, the gearbox takes a rotational input from the engine, converts this rotation by a given ratio and outputs the new rotation to the differential that then drives the front wheels of the car. The driver of the vehicle selects these ratios by moving a gear stick: it is standard practice to have a choice of five forward gear ratios and one reverse ratio. These are termed 'speeds', i.e. the first ratio is the first speed, the second ratio is the second speed, etc. This configuration is known as a '5-speed gearbox'.

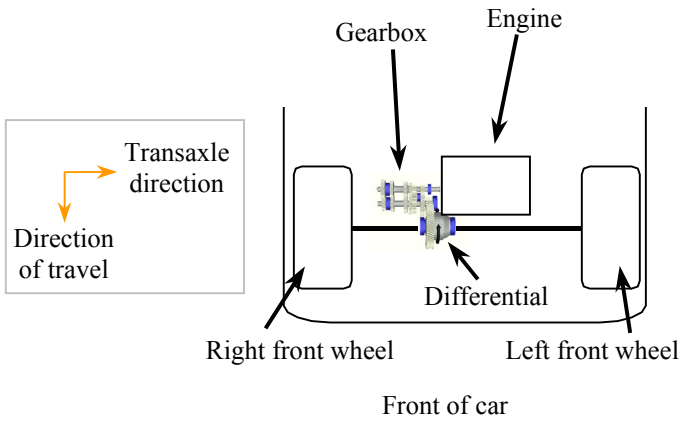

Figure 7-4: Sketch of front wheel drive passenger car layout, adapted from (James 2003)

As more powerful vehicle engines have become more affordable and popular, there has been a trend for gearboxes to have more speeds. Older cars, such as pre-1980s vehicles, mostly used 4-speed gearboxes. 5-speed gearboxes are now considered standard, while 6-speed gearboxes are also common (Arzethauser 2003), providing the driver with more ratios to allow optimal torque and power levels for varying driving conditions. Some high-powered vehicles, e.g. the Bugatti Veyron[37], have 7-speed gearboxes.

As the axis of the front wheels is restricted to lie across the vehicle and orthogonal to the direction of travel, it is common practice to align the shafts of the gearbox in this direction as well. This layout is termed a transaxle ('across the axle') gearbox (James 2003). The differential, required to permit different rotation speeds of the front wheels to allow for steering, is then aligned between the front wheels at the output end of the gearbox. The simplest layout that can be considered for a 5-speed gearbox is shown in Figure 7-5.

[37] The Veyron engine provides 736 kW at 6,000 rpm. http://www.bugatti-cars.de/bugatti/ (last accessed 9 January 2004).

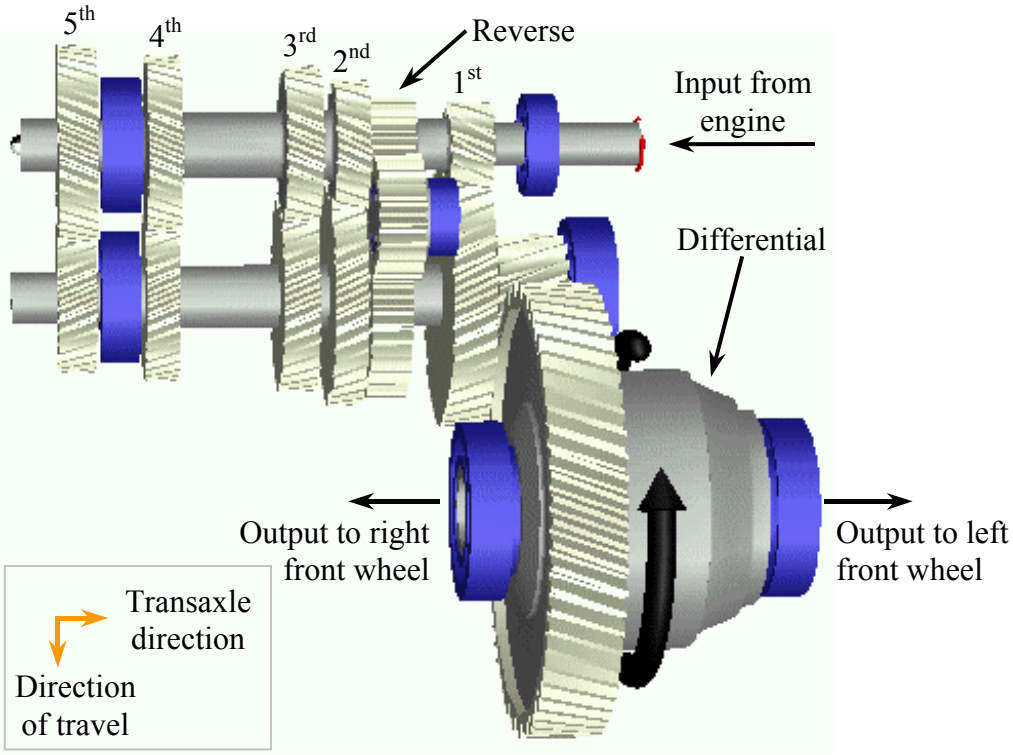

Figure 7-5: A standard 5-speed gearbox layout, adapted from (James 2003)

This standard transaxle 5-speed gearbox has three main shafts. The output shaft from the engine is connected to an intermediate shaft by a set of different gear pairs, one for each speed available to the driver. If no speed is selected, the gear disks on the intermediate shaft do not grip the intermediate shaft. Hence the engine can run with the input shaft rotating and all the gear pairs on the intermediate shaft spinning freely. This is termed 'neutral' speed. If the driver selects one of the forward speeds, a clutch mechanism activates to connect the intermediate shaft with the relevant gear pair. Hence an input rotation causes the intermediate shaft to rotate, resulting in power being transferred to the differential and thus also to the front wheels of the vehicle. Reverse gear works in a similar manner, except that there is another gear disk on a separate shaft that engages to rotate the wheels of the car backwards.

A main design issue with transaxle gearboxes is the space restriction between the engine and the outside of the vehicle, as the former is constrained to be centrally mounted. The gear pair for fifth speed in the standard layout considered in Figure 7-5 is quite close the right front wheel of the vehicle. A 6-speed gearbox based on this layout with another gear pair added to the end of the shafts would be difficult to

implement. Another difficulty is that long shafts flex when loaded, resulting in substandard meshing characteristics and reduced performance. Using a greater number of intermediate shafts for alternative layouts would result in greater complication but could resolve some of the issues with the standard layout.

Several alternative layouts already exist in current cars. The layout of a new design, used in the Rover 75 and other vehicles, is shown in Figure 7-6. The differential is shown without its protective covering as used in Figure 7-5, otherwise the differential design is similar. Third, fourth and fifth speed work in the same fashion as in the standard layout: the input shaft drives intermediate shaft A via the relevant gear pair and power is transferred to the differential.

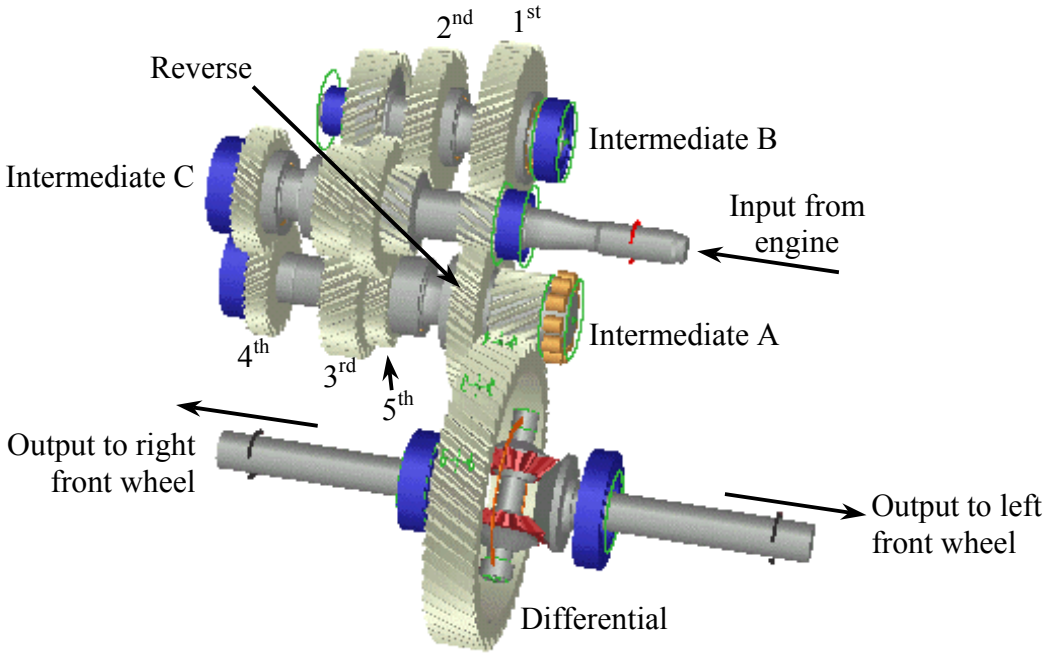

Figure 7-6: An alternative 5-speed gearbox layout, adapted from (James 2003)

First and second speed function in a different way. The input shaft actually consists of two concentric shafts, the original input axle that connects to the engine and a concentric sleeve (intermediate C). For third, fourth and fifth speed these are locked together and the fused entity acts as a single shaft, passing power from the engine directly to intermediate shaft A and then on to the differential. When first or second speed are selected, the concentric shafts (input and intermediate C) disengage, and

power flows between these concentric, non-fused shafts via intermediate shaft B. The third speed gear pair is then used to transfer power to intermediate shaft A. Figure 7-7 shows the alternative 5-speed gearbox layout when first speed has been selected. All non-loaded gears have been removed from this graphic. Note how only one gear on intermediate shaft C is used, hence it acts as an idler for this gear train.

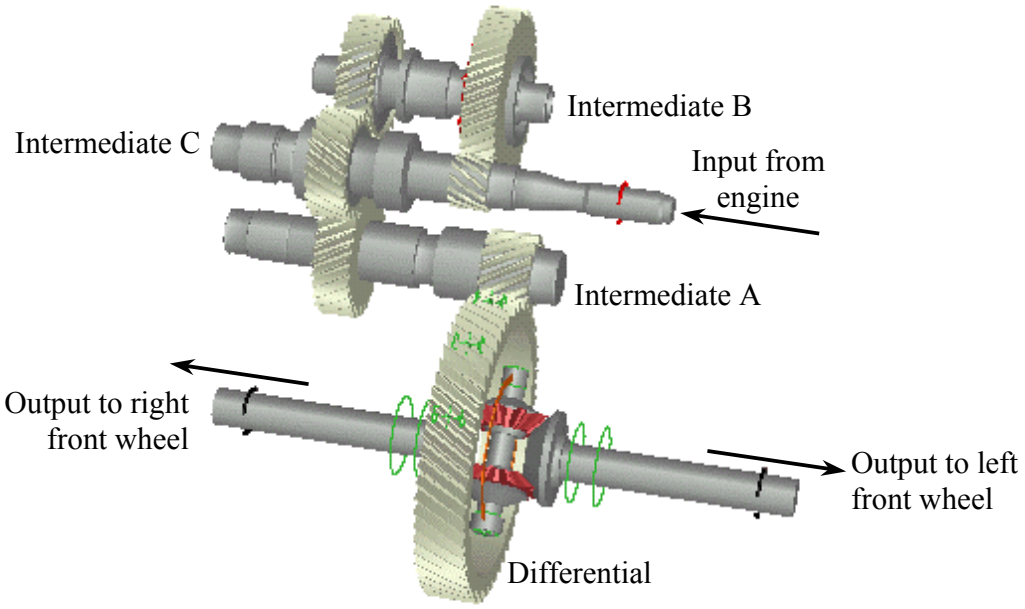

Figure 7-7: First speed loaded gears for alternative 5-speed gearbox layout, adapted from (James 2003)

The differences between the standard and alternative gearbox layouts are summarised in Table 7-1. There are situations when the standard gearbox layout is not adequate for the type of car being designed. In these cases, being able to generate possible alternative layouts suited to the particular new requirements would be beneficial. Generating a host of new design ideas could widen the possibilities considered by a designer and therefore increase the chances of finding a configuration that is tailored to the vehicle being designed.

Table 7-1: Summary of gearbox layout options

Gearbox layout	Advantages	Disadvantages
Standard	• Conceptually simple • Low part count	• Long shafts • Restricted gear ratios due to distance between shafts
Alternative	• More gear ratio options • Requires less space in transaxle direction	• Complex • Higher part count

This case study was carried out in conjunction with Romax Technology, a small company that produces gearbox design software. Their main product, RomaxDesigner[38], is used by many leading automotive firms to design gearboxes. The core competence of Romax Technology is the complete modelling of mechanical, specifically automotive gear-based, systems. RomaxDesigner models gearboxes to a high level of detail, including specifics such as addendum and dedendum values, and allows detailed optimisation of gear tooth profiles as well as other features. This attention to detail makes it possible to analyse complex phenomena such as shaft misalignment and gear whine, performance characteristics that are important to gearbox manufacturers.

A computational synthesis tool incorporating simulation-based evaluation using RomaxDesigner would enable the automated generation of new gearbox layouts. The alternative layout for the 5-speed gearbox introduced in this chapter is not difficult to visualise, however, other vehicles require more complex layouts. On-road heavy-duty trucks, such as Kenworth's T604[39], and off-road vehicles, such as tractors, feature gearboxes with up to 18 speeds. The design alternatives for such problems are numerous and a computational method could be of assistance to enable designers to rapidly explore a wide range of alternatives. A detailed modelling tool like RomaxDesigner complements the higher-level gear system representation used by the parallel grammar.

[38] http://www.romaxtech.com/ (last accessed 2 December 2003)
[39] http://www.kenworth.com.au/t604/summary.asp (last accessed 2 January 2004)

This case study is of particular relevance to the automotive industry. It is generally thought that current vehicle transmission technology requiring gearboxes will be used until at least the year 2020 (Poon 2003). Computer modelling and simulation in this design domain has been increasing in recent years and the incorporation of computational synthesis is considered the next step towards faster gearbox development and implementation cycles. Computational synthesis is thus of strategic importance to the vehicle industry (Poon 2003).

7.2. Clutches

Before the case studies can investigated, the existing parallel grammar requires one main extension: this is necessary as the design representation used for the clock and camera case studies does not support the connection of two shafts by multiple gear pairs. Such arrangements occur in gear mechanisms where different speeds can be selected (c.f. Figure 7-5).

No changes to the structure grammar are required for multiple connections, as clutched configurations can be created using the existing parallel grammar C-Rules. This process is demonstrated in Figure 7-8. The sequence shows structure rule 2 being applied to add a new shaft to an existing partial design. Subsequently, two separate applications of rule 3 are carried out that add two gear pairs of differing ratio between the new shaft and the rest of the mechanism.

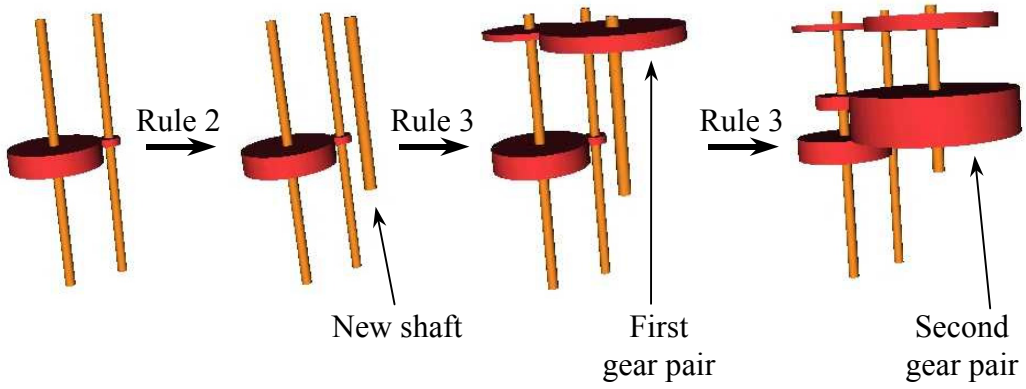

Figure 7-8: Multiple shaft connections with existing structure rules

Multiple gear pairs connecting the same shafts cannot be engaged at the same time, as this would cause the system to be over-determined, so no movement would occur. Clutches are used to mesh gear pairs in turn, whereby the gear disks are splined[40] and can be moved axially over small distances to engage with the shaft at their centre. Detailed component representations of these clutches are not included in the design representation as they are considered beyond the scope of parametric synthesis for conceptual design stages.

While the structure grammar remains unchanged, the function grammar requires an additional rule to enable multiple gear connections between axles to be represented. The current set of function rules (Figure 5-5) does not allow creation of a new edge between vertices with an existing connecting edge. A new rule is added to allow such a change, resulting in a multigraph representation that allows more than one edge between the same nodes. The new function rule is shown in Figure 7-9. The set of labels used has been changed to reflect the new components used in the drill and gearbox case studies. These are listed with a short explanation in Table 7-2.

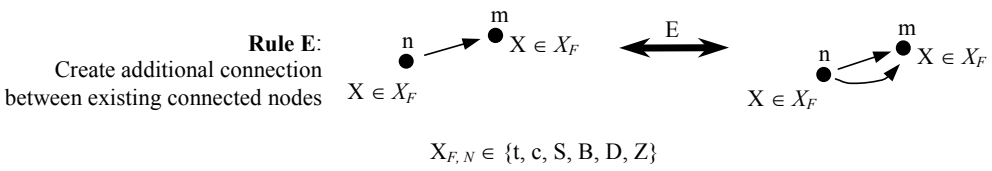

Figure 7-9: New function rule E for the drill and gearbox case studies

[40] A spline is a key that fits into grooves in a shaft, wheel or other attachment so as to allow longitudinal movement of the latter (source: Oxford English Dictionary).

Table 7-2: Explanation of terminal labels used for the validation case studies

Label	Meaning
c	A terminal vertex indicating further connectivity (no change from previous definitions).
S	A terminal vertex that corresponds to the input shaft.
B	A terminal vertex that corresponds to the drill bit output shaft.
D	A terminal vertex that corresponds to the differential output shaft. The differential and front wheels are attached to this shaft.
Z	A terminal vertex that corresponds to a shaft that has a separate behavioural mode whereby it can be rigidly connected to a concentric shaft through clutch activation.

7.2.1. Black and Decker drill

The updated parallel grammar can now be used to investigate the industrial case studies presented. A function representation of the corded drill is created using the parallel grammar. This is shown in Figure 7-10, using the same triangular grid to lay out the graphs as in previous examples. Two possible sequences of rule applications are shown. The sequence to the right of the figure shows a set of rule applications that builds a path to the output node in small steps from the input. The left hand sequence uses rule D to insert the connection node after a direct edge from input to output has been created.

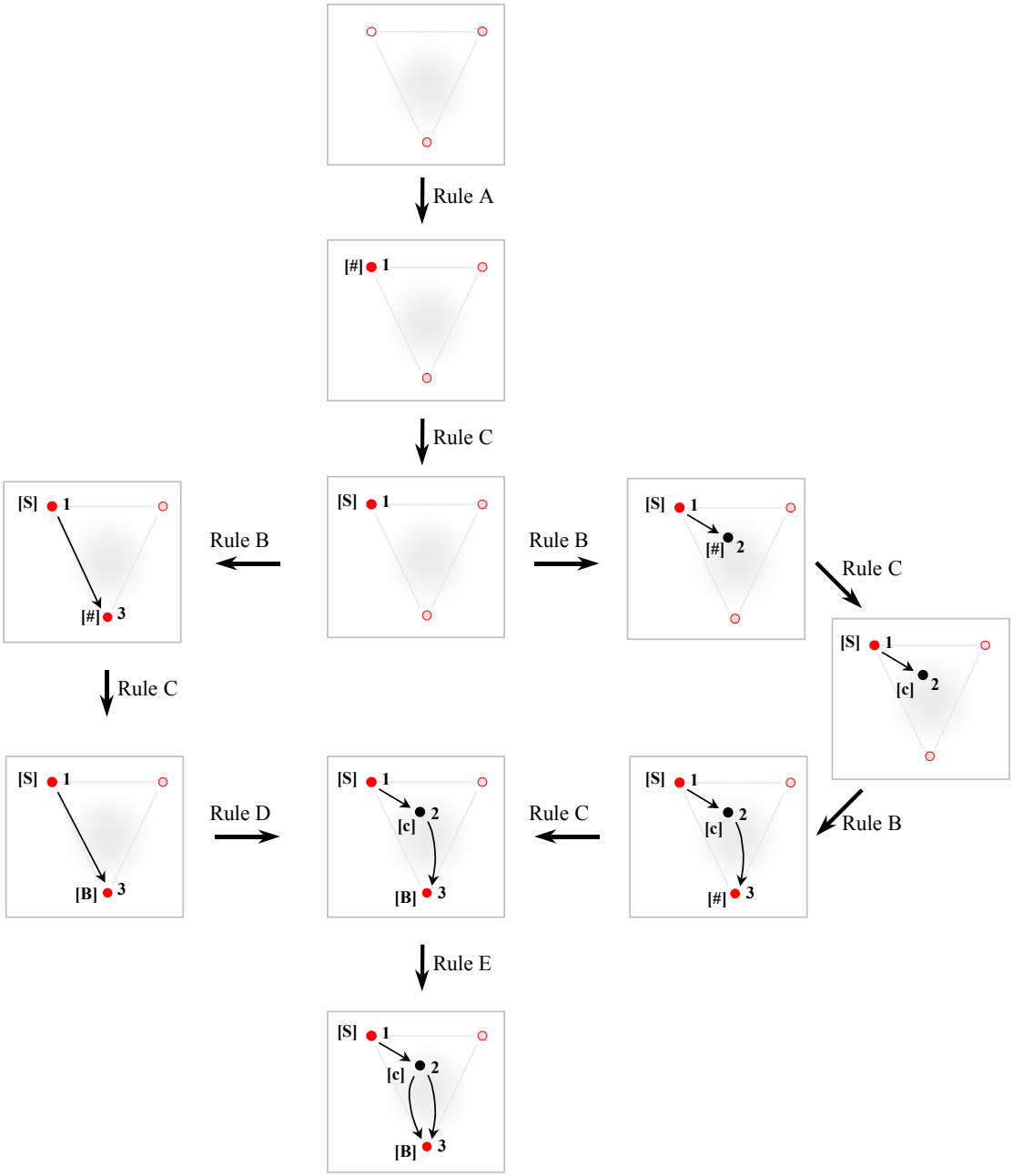

Figure 7-10: Two possible rule sequences to create a function representation for the drill case study

The final function graph in Figure 7-10 is a directed acyclic multigraph as it allows more than one edge to connect the same two vertices. The function graph at the end of the generation sequence corresponds to the structure representation produced in Figure 7-8. The final parallel representation is shown in Figure 7-11. Models generated can then be used to explore design possibilities using evaluation data from geometry-based metrics and from simulation data as demonstrated in chapter 6.

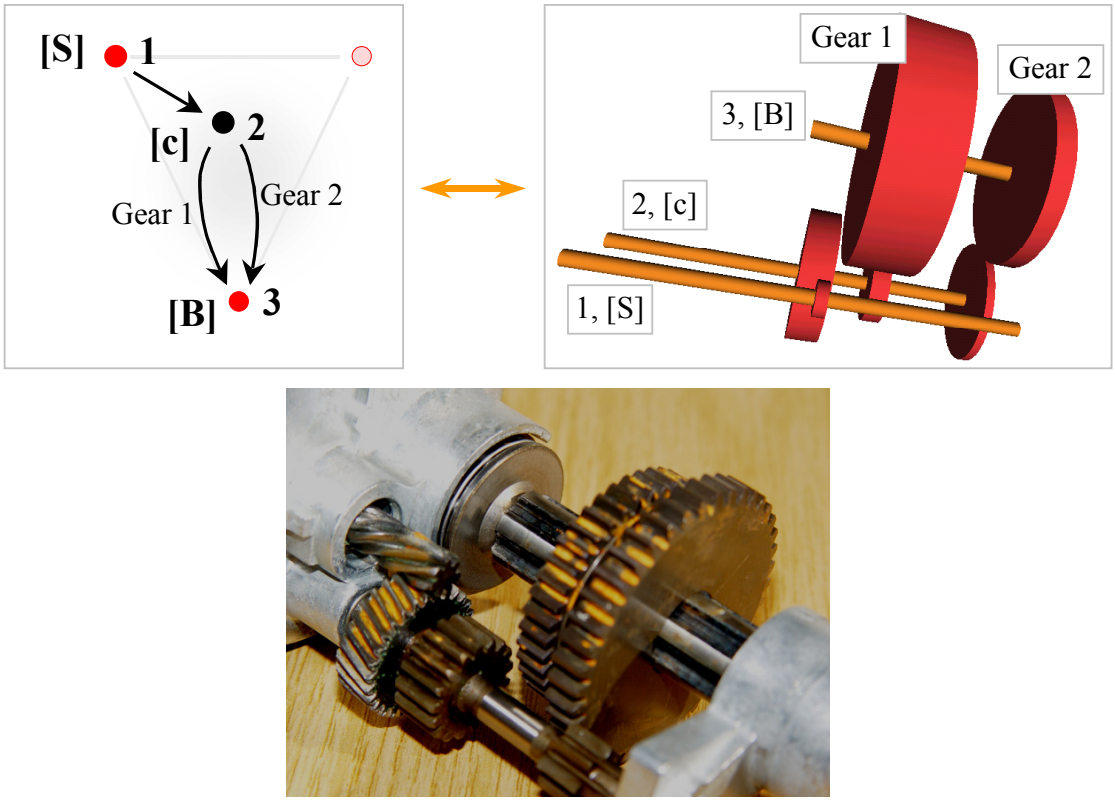

Figure 7-11: Drill case study: function (top left) and structure (top right) representations; assembled mechanism (bottom)

Assembled mechanism is shown in neutral state, i.e. neither gear is engaged.

As shown in Figure 4-7, the function graph can be thought of as representing power flow through the design. The function graph for the drill case study, created in Figure 7-10, has two edges connecting vertices 2 and 3. However, only one edge can be activated at any one time using a clutch mechanism, otherwise the one degree of freedom mechanism is overconstrained. The two power flow paths for the two drill speeds are unambiguous in this graph.

The corded drill case study is a useful introduction to the multigraph representation. Design improvements based on variations of the existing design shown in Figure 7-10, such as the introduction of a new shaft to allow smaller gear disks to be employed, would most likely be outweighed by an unacceptable increase in part count. The drill layout is unlikely to change from its existing simple layout. The design challenge for this particular product is in providing performance such as reliability, high power and aesthetics at very low cost.

7.2.2. Transmission system

The case for a computational synthesis approach to design generation is more compelling for the vehicle transmission case study. The alternative 5-speed gearbox in Figure 7-6 has been successfully used in current vehicles to fit the gearbox into a constricted space, thus solving a major design issue. The growing demand for 6-speed gearboxes, where these layout problems are particularly acute, indicates that a computational method of generating new designs, i.e. exploring different design clans, could be of real advantage. Furthermore, off-road and heavy-duty vehicle gearboxes, which can have eighteen or more speeds, are even more complex and present further challenges. A computational method of exploring the multitude of valid design possibilities could be effective for generating alternative performance-driven designs.

A graph representation of the standard 5-speed gearbox configuration (Figure 7-5) is shown in Figure 7-12. This graph is relatively straightforward, containing five separate paths between vertices 1 and 2, corresponding to the five speeds of the gearbox, and a single connection between nodes 2 and 3, corresponding to the output to the differential.

Figure 7-12 also shows a representation of the alternative 5-speed gearbox layout (Figure 7-6). A new label, [Z], is introduced, corresponding to concentricity with its parent vertex. This new label represents the dual function of such concentric shafts, i.e. they can be coupled and decoupled using a clutch mechanism. The edge between nodes 1 and 4 is dashed to highlight the special status of this connection, showing that it does not correspond to a standard gear pair but to a connection between concentric shafts in the structure representation, e.g. input shaft and intermediate shaft C in Figure 7-6.

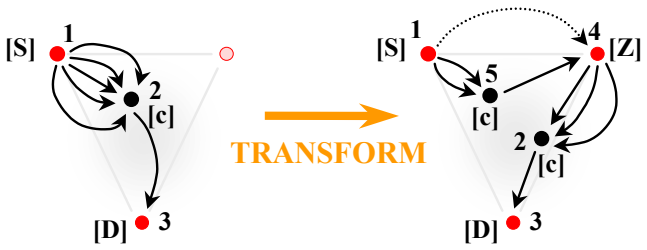

Figure 7-12: Function graph for standard (left) and alternative (right) 5-speed transaxle gearbox layout

The representations in Figure 7-12 are similar to existing graph grammar approaches, e.g. (Schmidt et al. 2000). If it were possible to perform a transformation on the standard 5-speed gearbox representation to enable the generation of the function graph of the alternative design representation, similar transformations could be used to generate further design clans that fulfil the same functional specification as the initial standard design. This work can be seen as a functional analogy to grammar transformation to follow style variations in architectural design (Knight 1994). Here the aim is to capture the language of gearbox designs that can be used for vehicle applications. The resulting function graphs from this exploration could then be used as an input for the parallel grammar to search for preferred designs.

Before this transformation can be studied in more detail it is necessary to distinguish between ‘active’ and ‘inactive’ edges in the graphs in Figure 7-12. In the simple drill case study there are two speeds that can be selected by the user and the two corresponding power flow paths in the function multigraph are unambiguous (Figure 7-10). In the 5-speed gearbox, however, there are more possible combinations of power flow paths than there are speeds. The five actual power flow paths are shown in Figure 7-13. Active edges, i.e. gear pairs that have been engaged, are coloured black and inactive edges, i.e. gear pairs that are not engaged, are shaded grey. Each separate gearbox speed can therefore be recorded by noting the sequence of edge traversals between input and output. The notation lists the sequence of vertices from input to output for each speed. If an ambiguous path is specified, i.e. there is more than one edge between two particular vertices, the vertex distinguishing identifier is noted in brackets. These sequences are included in Figure 7-13 for the alternative 5-speed gearbox configuration.

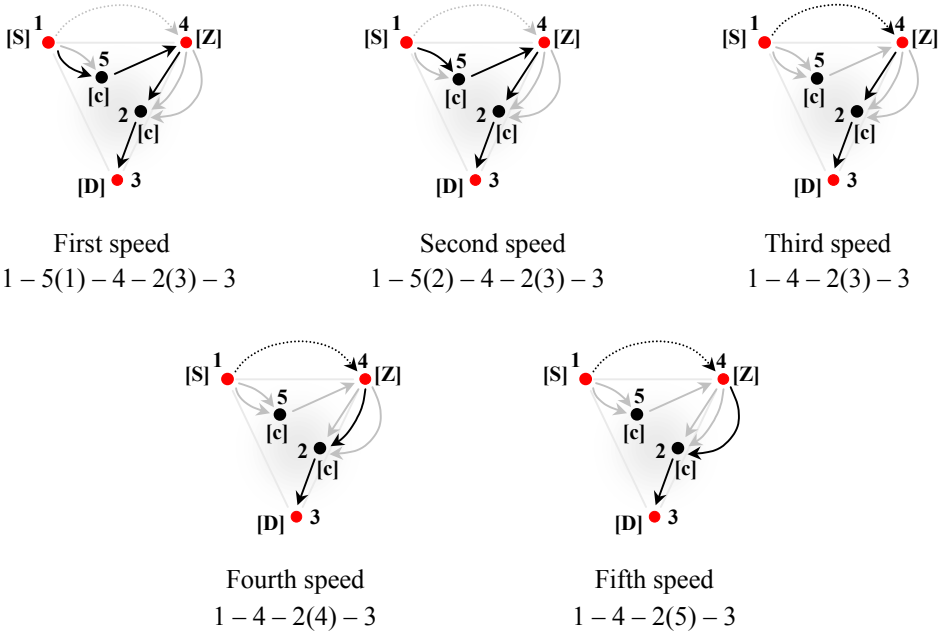

Figure 7-13: Active edges for different speeds in the alternative 5-speed gearbox graph (RHS of Figure 7-12) with node sequence for each speed

(active edges black, non-active edges shaded grey)

7.2.3. Graph modification

The graph grammar has not yet been used to computationally generate designs from different design clans. For the gearbox case study, two existing designs have been considered, a standard and an alternative 5-speed configuration, both taken from current industry examples. The former is a common, simple layout used historically since the advent of the motorcar[41]; the latter is a more recent innovation. Capturing the changes required to modify the graph representation of the standard gearbox to create that of the alternative gearbox would enable the same transformation rules to be used to discover other new design configurations.

To find new gearbox configurations, graph exploration is attempted separately from the parallel grammar environment, as linking all graph changes to corresponding structure changes initially could hinder the process. The main aim is to consider

possible successful design clans, therefore consideration of architecture, i.e. design families, and detailed design parameters will not be considered here.

Further function rules are required to fully capture the design transformation illustrated in Figure 7-12. The two function rules in Figure 7-14 allow function graphs to be altered more radically, enabling the detachment and reattachment of existing graph edges. For the purpose of representing a parallel set of one degree of freedom mechanisms, such as a gearbox, these new modification rules may not be used to make modifications that introduce circuits into the graphs. Circuits represent overconstrained mechanisms and therefore an exploratory move that creates such a circuit must be considered illegal.

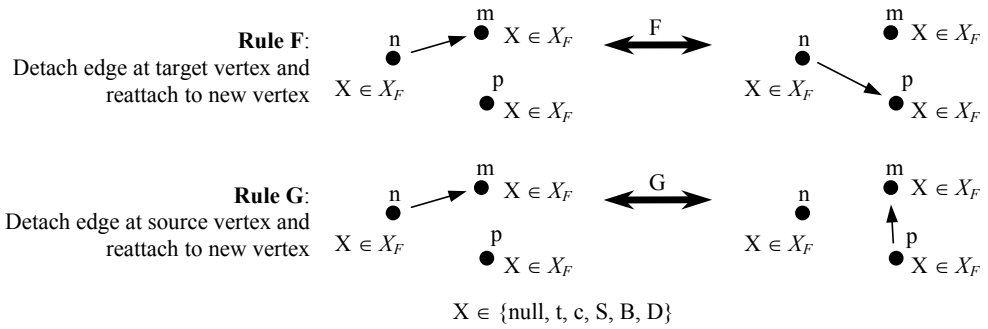

Figure 7-14: Graph modification rules

An example sequence of legal exploratory rule applications is shown in Figure 7-15 to transform the graph representing the standard 5-speed gearbox into the graph representing the alternative configuration (Figure 7-12).

[41] French carriage-makers Panhard and Levassor are credited with the first use of clutched gears on two parallel shafts in the late 19th century, replacing previous belt-driven drive mechanisms. http://www.citroen.mb.ca/citroenet/html/p/panhard1.htm (last accessed 16 January 2004)

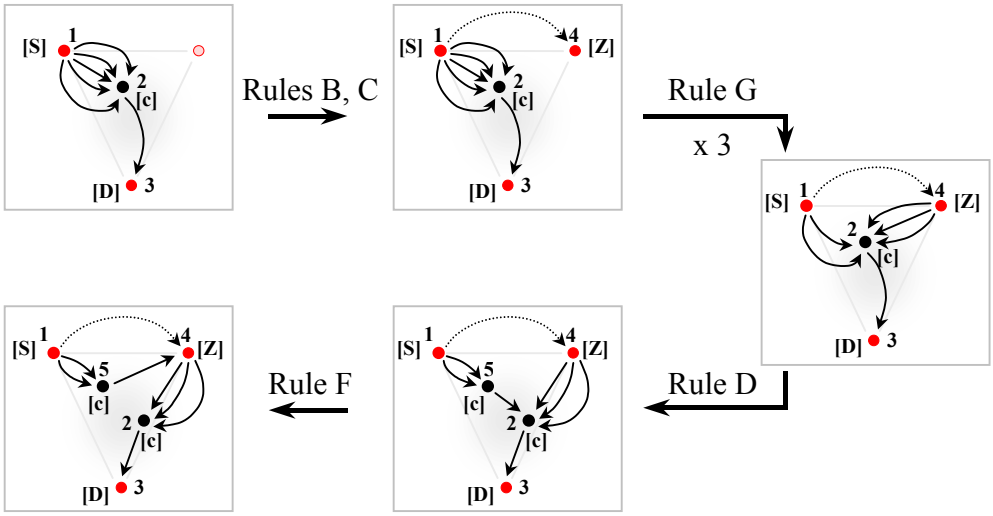

Figure 7-15: A sequence of exploratory function graph transformations.
The initial (top left) and final (bottom left) graphs are the standard and alternative 5-speed gearbox configurations from Figure 7-12.

The framework provided by the graph modification rules allows the creation of further configurations. An example of such a design, generated using the grammar rules by hand, is shown in Figure 7-16 with active edges for each speed shown in Figure 7-17. This layout has three concentric shafts, nodes 1, 3 and 5. Such a complex layout may well not be viable for a production vehicle, however, such an arrangement could be applicable if a closely packed transmission is required.

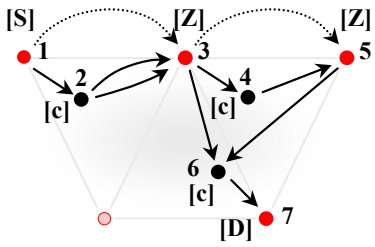

Figure 7-16: 5-speed gearbox configuration synthesised with the exploratory grammar

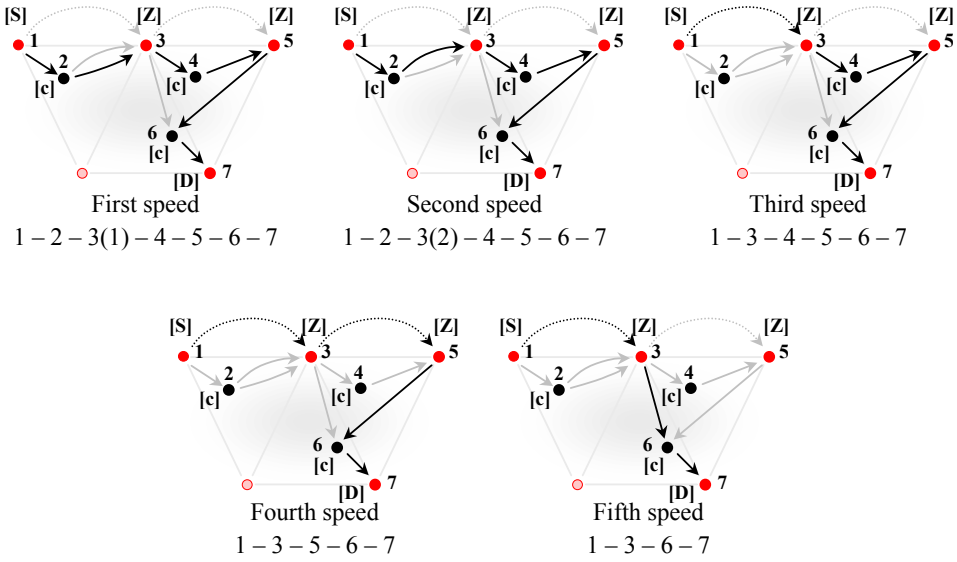

Figure 7-17: Active edges for different speeds in the novel 5-speed gearbox graph (Figure 7-17) with node sequence for each speed to indicate power flow

(active edges black, non-active edges shaded grey)

A grammatical approach is equally applicable for more complex redesign problems. Off-road vehicles, e.g. trucks, construction vehicles and piste-bashers[42], are often equipped with gearboxes that provide high torque for adverse terrain. Such gearboxes require more speeds than standard on-road vehicles and transmissions with an excess of 15 speeds not being uncommon. The 15-speed gearbox shown in Figure 7-18 is based on a current design[43] on the market with a 2.2 kNm maximum torque specification. Power flow diagrams for each speed are shown in Figure 7-19. The five high torque speeds are referred to as 'deep reduction' (DR) gears.

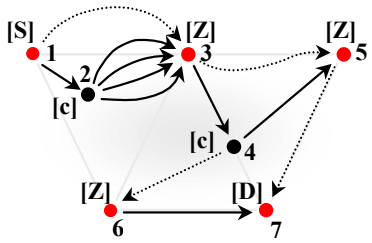

Figure 7-18: Function graph for 15-speed gearbox for on/off-road vehicle

[42] Piste-bashers are tracked vehicles with large snow shovels that are used to prepare flat and safe areas on mountainsides (referred to as pistes) for people to ski on. A large range of gear speeds enable vehicular access to steep mountain slopes as well as allowing high speed travel over flatter terrain.
[43] http://www.roadranger.com/csee/trans_srvman.htm (last accessed 29 December 2003)

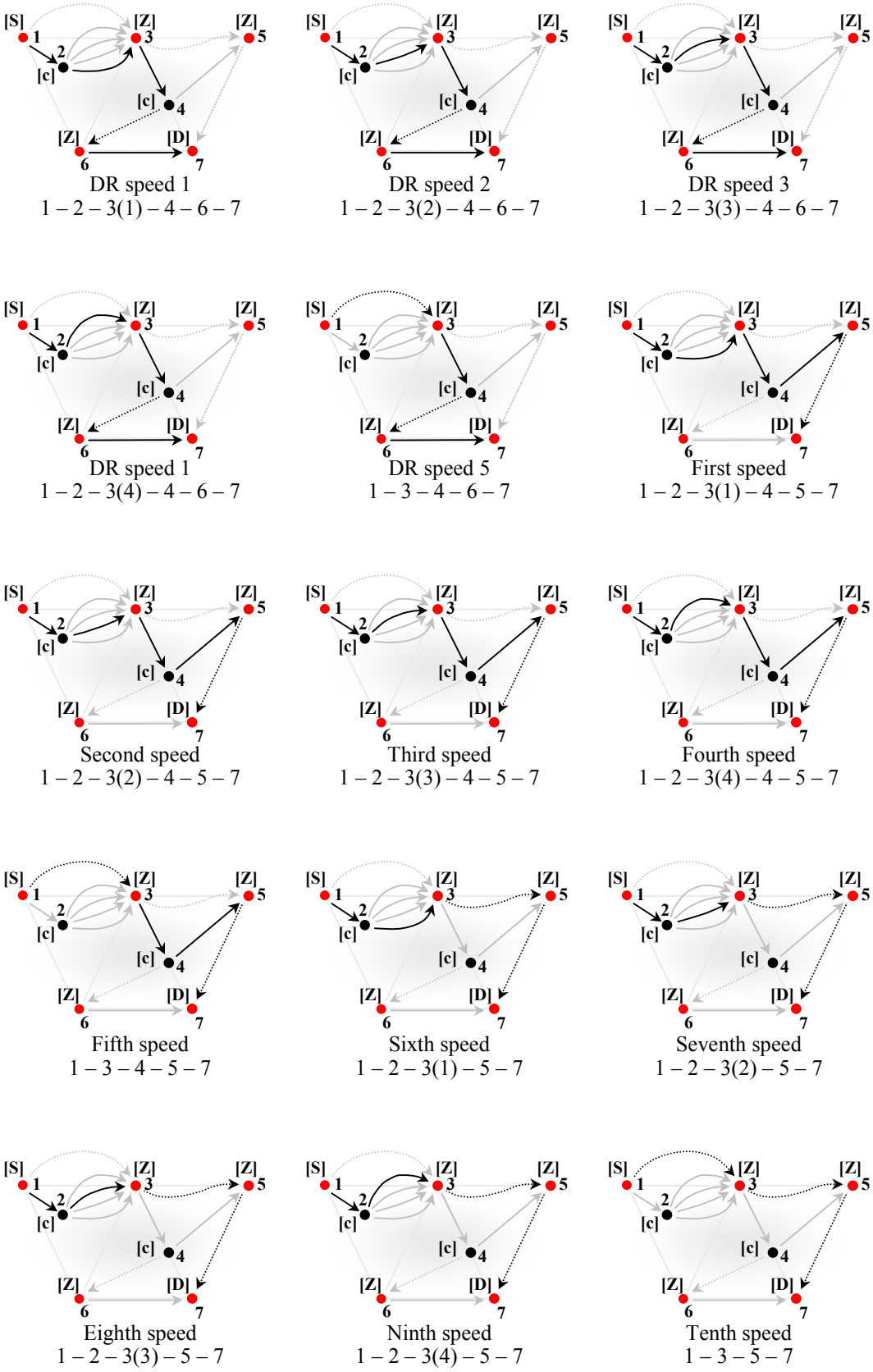

Figure 7-19: Active edges for different speeds in the 15-speed gearbox graph with node sequence for each speed to indicate power flow

(active edges black, non-active edges shaded grey)

For large trucks that use many-speed gearboxes, the space constraints that made the alternative 5-speed gearbox layout necessary (Figure 7-6) are not usually quite so pressing. However, alternative layouts can still be advantageous. The high levels of torque required by off-road applications means that shaft deflection is an increased risk, to the extent that the structure of such designs is split using mirrored off-axis intermediate shafts to decrease the load on individual components (see schematic in Figure 7-20). Hence reducing space is a driving constraint even in these large-scale designs.

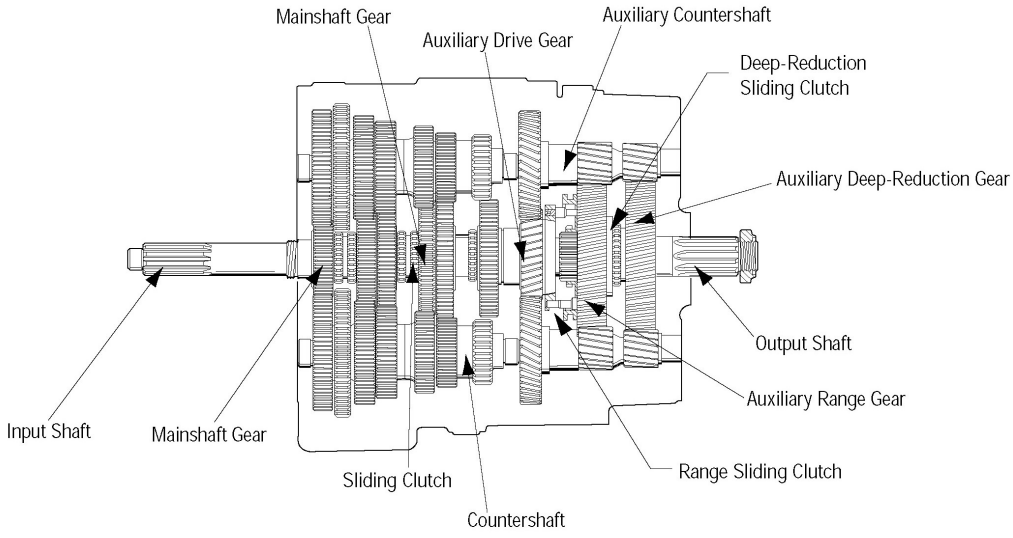

Figure 7-20: Schematic layout of 15-speed gearbox[44]

A function graph for an alternative 15-speed gearbox layout that can be generated using the extended function grammar is shown in Figure 7-21. A gearbox based on this configuration is shorter with the mainshaft gears corresponding to vertex 8 aligned above and below the plane of the diagram in Figure 7-20.

[44] Eaton Fuller Heavy Duty Transmissions, Service Manual TRSM-1500, February 2003

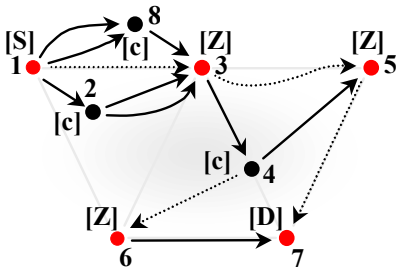

Figure 7-21: Alternative layout of 15-speed gearbox

7.3. Conclusions

This chapter has investigated the industrial applicability of the parallel grammar for computational synthesis of gearboxes. The method is validated through analysis of industrial case studies that demonstrate the parallel grammar and show its industrial relevance. The method is used to investigate corded drill and vehicle transmissions designs, specifically 5-speed transaxle gearboxes. These case studies are more complex than previous examples given for two reasons. Firstly, power flow through these designs is considerably higher than in the previous examples. Secondly, these case studies contain multiple possible power flow paths, controlled by clutches that engage and disengage gear pairs, to provide different operating speeds. The latter required an extension to the function grammar to represent and generate clutched gear systems.

Different design clans, in this case alternative 5-speed gearbox configurations, are created using a modification grammar, based on the existing function grammar, that transforms existing graphs by manipulating edges to create new connections. This new grammar allows the transformation of a function graph representing the standard 5-speed gearbox layout into a graph representing the alternative layout. Further design configurations, such as those in Figure 7-16 and Figure 7-21, can also be generated with this modification grammar.

8. Discussion and conclusions

This dissertation has investigated the development of a computer-based method of synthesising solutions to mechanical engineering design problems. The method uses a parallel grammar that builds up and modifies designs using dual representations of function and form; constraints ensure that designs created by the parallel grammar are topologically and parametrically valid. Performance evaluation data, derived from geometry-based metrics and automatic behavioural simulation, is used to create single and multiple objective functions enabling stochastic search algorithms to find preferred performance-driven designs.

Verification for this work is carried out by reproducing existing designs for case studies based on a clockwork clock, a simple mechatronic camera, a corded power drill and manual gearboxes for on- and off-road vehicles. The synthesis tasks in these case studies centre around the creation of serial and parallel gear trains using parametric components. Validation of the research is demonstrated by assessing the strategic needs of an industrial firm that specialises in design and modelling of automotive vehicle transmissions. Based on initial results, the possibilities for synthesis of new designs – using an exploratory grammar and subsequent embodiment using the parallel grammar framework – are shown to be of relevance to industry.

8.1. Contributions

The aim of this research, captured in the thesis statement (Figure 1-1), is to prove the viability of methods for computational synthesis in mechanical engineering design domains. This research investigates the feasibility of such synthesis methods using a parametric synthesis framework (Figure 2-3).

A summary of the main contributions of this thesis is now presented:

- A parallel grammar is developed within the parametric synthesis framework to generate families (chapters 4 and 5) and clans (chapter 7) of designs using a parallel representation of both function and parametric form. This grammar, implemented for four case studies (chapters 4 - 7) using common grammar rule libraries with domain-specific rules added as necessary, is shown to be useful in aiding the creation of novel design configurations and generating parameters for virtual prototypes. The parallel grammar fulfils the three criteria that allow characterisation as a design compiler (Ward 2001), c.f. section 3.2.

- The parallel grammar uses topological and parametric constraints to ensure validity of the design solutions generated so that they match the task specification (chapter 4). The use of collision detection libraries for constraint checking enables an effective approach to three-dimensional packing problems, allowing connected structures to be built up from scratch (chapter 4) and subsequently modified (chapter 5) to generate preferred designs.

- Mechanical performance includes both behavioural and spatial requirements. Performance-based synthesis is pursued through the use of geometric design metrics (chapter 5) and the use of behavioural modelling (chapter 6). To direct the search for preferred designs, automatically generated simulation models are analysed throughout the synthesis process to provide feedback on the quality of designs as they are generated.

- A new multi-objective hybrid pattern search algorithm (chapter 6) is developed to add a pragmatic approach to stochastic search in cases when evaluation is computationally expensive.

- An appraisal of the viability of parametric synthesis based on a parallel grammar is provided by two industrial case studies (chapter 7). The parallel grammar shows particular promise for use in the design and development of many-speed novel vehicle transmission configurations.

8.2. Discussion

The C- and P-Rules of the parallel grammar enable the generation and parametric modification of designs, providing the means to explore a vast design space. The size of this design space, i.e. the design language, made accessible to algorithm-driven exploration by the parallel grammar, results in combinatorial explosion. Geometric and topological constraints bound the search space in order to focus on purposeful designs and, coupled with behavioural design evaluation, ensure validity of designs generated by the grammar.

The parallel representation of function and form used by the synthesis framework enables the differentiation of simple rules to maximise generative flexibility using the grammar while still allowing the representation of complicated structures. The parametric synthesis method enables a bottom-up approach to design, piecing together and modifying components to fulfil a previously established specification. The representation used to describe designs is compatible with behavioural modelling techniques that can be used to apply domain-specific knowledge to design problems initially investigated with a more general approach. For example, the parametric camera gear train designs generated in chapter 6 were exported to a more sophisticated modelling environment to provide simulation data in the particular environment of the camera model.

This bottom-up nature of generating designs using the parallel grammar has consequences that affect the success of the synthesis formalism. The application of parallel grammar rules requires determination of application points, i.e. where in the design to apply a rule, and input parameters. When used in conjunction with automatic synthesis methods, these parameters are generated stochastically. The success of grammar rule application depends on suitable input, but it is often difficult to determine in advance what parameter ranges and application points are likely to be more successful than others. The incorporation of machine learning techniques, e.g. (Vale and Shea 2003), might be used to increase the success rate of parametric modification, thereby improving the effectiveness of search for optimally directed

designs. An alternative method of increasing the success rate of rule applications would be to incorporate more knowledge into grammar rules, but this could unduly restrict the design space and thus would not necessarily be a good approach.

The systems approach of the parametric synthesis method is suited to the case studies analysed in this research, where specification of design synthesis tasks is based on input/output characteristics. Particular constraints are added to these basic requirements by a user. Design tasks can be achieved in modular fashion, for example one could synthesise automotive gearbox designs and subsequently generate solutions for a differential to be added to the vehicle design. However, when breaking down design problems into small blocks, as with the bottom-up approach described here, existing concepts of modularity are not always adequate for representation of complexity. The use of behavioural simulation for evaluation of designs as they are generated enables the evaluation of this complexity in the form of emergent behaviour, i.e. behaviour that is more intricate than was initially envisaged.

Behavioural evaluation of a design requires the existence of a simulation model that represents the mechanism under consideration. The current method requires this model to be made by hand, separately from the computational synthesis process. The parallel grammar contributes only the parameters for an already-specified family of designs. Generating such simulation models is complicated: capturing the functionality of the camera in the model is a difficult task. Once completed and pre-compiled, the models require long evaluation times (c. 15 s) that are not desirable for a synthesis tool capable of rapid generation of new designs. In chapter 6, search is hampered by a slow evaluation time for behavioural performance metrics. Two areas, therefore, need addressing: (1) automatic creation and (2) faster evaluation of simulation models.

Scalability issues are important in design synthesis research, as solvable small-scale problems can become intractable on a larger scale. The initial generation of designs using the parallel grammar is susceptible to scaling, as adding more components does, in general, result in structures with more complex constraints. Once initial valid designs have been generated, however, modification using perturb rules takes place on as local a basis as required by the specific grammar rule that is in use. Scaling has less

of an effect on the success of this process at a local level, i.e. whether modifications are successful or not, but search times are still affected due to consideration of a greater number of design variables.

8.3. Further research

There are many avenues of possible future work that could make a mature technology for mechanical design from the synthesis formalism explored in this thesis. These avenues have been separated into two groups. Firstly, short term work is outlined, listing areas of research that could be taken further to develop the research presented in this dissertation. Secondly, longer term research goals are posited, indicating how specific projects could take this research into new areas to explore related topics.

8.3.1. Short term

The parallel grammar has been implemented for connected systems using components such as shafts, spur gears and support plates. This work could be extended to use a larger library of parts to enable more design domains to be considered. As an example, the inclusion of bevel gears would enable the analysis of gearboxes based on layouts other than the transaxle designs considered in chapter 7.

The parallel grammar supports generalised layouts, but further work is required to unleash this generality in the current implementation. The data structures used for the case studies in this research constrain shafts to lie parallel to each other. The incorporation of new components for the solution of other problems, such as longitudinal gearbox layouts, requires shafts to be aligned parallel to the direction of vehicle travel.

Further work is required to enhance the hybrid pattern search to record more information about successful grammar rule applications, such as parameter ranges and points of application, to learn modification strategies that might improve convergence on optimally directed designs during search.

The current behavioural modelling process using Dymola could be improved by identifying key areas of a design that need to be modelled and then focussing on these alone. In chapter 6, the current implementation models the complete camera mechanism each time an evaluation is carried out and hence a reduction in the size of the model could lead to a faster evaluation time. The difficulty in making such model-reducing decisions lies in deciding what architecture chunks are critical.

8.3.2. Long term

The automatic behavioural evaluation method used in this thesis is an exciting development in the field of synthesis and could form the basis of a dedicated project that would focus on generating more fundamental behavioural analysis models 'on the fly'. The camera simulation model in chapter 6 requires pre-compiled executables for each family of designs that are to be considered. Extending this technology to be able to create pre-compiled executables that can be used for each clan of designs would enable a greater variety of designs to be evaluated for behavioural performance.

Due to the nature of the parallel grammar, the form and function representations used are straightforward. More sophisticated, yet still simple representations, e.g. based on features, might be used to allow the parallel grammar to develop more complex designs without introducing undue complexity to the grammars used.

The parallel grammar has demonstrated potential for use in the specific area of vehicle gearbox design. A dedicated project could see this work implemented in more detail, with grammar rules, constraints and modelling techniques tailored to this particular design field.

8.4. Final words

True performance-based computational synthesis is an exciting prospect for mechanical design. It is hoped that, with a view to underpinning future developments and advances, the research presented here has helped piece together those parts of the synthesis jigsaw puzzle that are of particular relevance to mechanical systems design.

9. Glossary

Explanation of terms and acronyms. Cross-reference indicates main or first text occurrence. Italics signify entry elsewhere in glossary.

AC	**A**lternating **C**urrent.	p. 167
Addendum	The maximum outward extension of a gear tooth profile from the *pitch radius* of that gear.	p. 76
A-Design	**A**gent-based **D**esign. An agent-based approach to *design synthesis* (Campbell et al. 2003).	p. 37
Back Loading	The interaction of modules in a design at run-time. Primarily of importance when performance of other modules is affected, e.g. loading an engine alters the behaviour of this engine.	p. 35
Bond Graphs	An method of representing physical systems by modelling energy interchange.	p. 52
CAD	**C**omputer **A**ided **D**esign. Commonly used to refer specifically to computer-aided sketching tools. See also *CaeD*.	p. 17
CaeD	**C**omputer-**a**ided **e**ngineering **D**esign. The use of computers in *design*, including, but not limited to *CAD*.	p. 24
CaeDRe	A **C**omputer-**a**ided **e**ngineering **D**esign **R**esearch **e**nvironment proposed by Bracewell and Shea (2001).	p. 25
CAM	**C**omputer **A**ided **M**achining.	p. 19
CEM Design	**C**omplex **E**lectro-**M**echanical **Design**.	p. 35
Clockwork	Mechanical mechanisms based on springs, ratchets	p. 67

	and winding gears.	
CPU	**C**entral **P**rocessing **U**nit.	p. 17
C-Rule	**C**reate **Rule**. A grammar rule that builds up a new part of a structure, i.e. creates a new part of the structure. See also *P-Rule*.	p. 75
CSP	**C**onstraint **S**atisfaction **P**roblem.	p. 58
DC	**D**irect **C**urrent.	p. 167
Dedendum	The minimum inward extension of a gear tooth profile from the *pitch radius* of that gear.	p. 76
Design	The deliverables of the *engineering design* process. See Definition 3-5.	p. 47
Design Synthesis	The process of making, generating or creating new *designs*. See section 3.2 for detailed discussion.	p. 34
DFA	**D**esign **F**or **A**ssembly.	p. 54
DFM	**D**esign **F**or **M**anufacture.	p. 54
DR Gear	**D**eep **R**eduction **G**ear. Refers to high torque speeds of a 15-speed *gearbox*.	p. 56
Dymola	A behavioural modelling tool. Dymola uses Kron's method of 'tearing' (Kron 1963) to solve sets of simultaneous equations.	p. 55
EDS	**E**dinburgh **D**esigner **S**ystem.	p. 50
EGT	**E**picyclic **G**ear **T**rain. A gear train with 'planetary' gears that rotate about a 'sun' gear.	p. 46
eifForm	A general structural engineering tool for computational truss *design synthesis* (Shea et al. 2003).	p. 43
Engineering Design	The process of engineering design. See Definition 3-1.	p. 29
Escapement	A type of ratchet device that controls mechanism advancement in a clock to enable time keeping.	p. 73
FBS model	**F**unction-**B**ehaviour-**S**tructure **model**. A modelling framework proposed by Umeda et al. (1990).	p. 48
FFREADA	**F**unction to **F**orm **R**ecursive **A**nnealing **D**esign **A**lgorithm. Design synthesis approach using	p. 45

	string grammars (Schmidt and Cagan 1998).	
FuncSION	'Compositional synthesis' approach to conceptual design using pruned exhaustive search (Liu et al. 2003).	p. 38
GA	A **G**enetic **A**lgorithm. A stochastic search algorithm that 'evolves' populations of solutions using natural selection.	p. 57
Gearbox	A part of a *transmission* system that contains a set of parallel gear trains of differing ratio. One of these can be selected at any one time to transfer power from engine to point of power application.	p. 20
GGREADA	**G**raph **G**rammar **R**ecursive **A**nnealing **D**esign **A**lgorithm. Successor to *FFREADA*, developed to synthesise designs that feature non-serial function and form dependencies (Schmidt and Cagan 1997).	p. 45
Grammar	A type of *production system*.	p. 39
Grammar Rule	Part of a *grammar*.	p. 39
GUI	**G**raphical **U**ser **I**nterface.	p. 45
Heuristic	A 'rule-of-thumb'.	p. 57
Heuristic Search	See Definition 3-8.	p. 57
HVAC	**H**eating, **V**entilation and **A**ir-**C**onditioning.	p. 59
Hybrid pattern search	Non-deterministic search method developed for use with the *parallel grammar* that, in a manner similar to conventional *pattern search* methods, seeks to repeat successful sequences of design modifications to increase search efficiency.	p. 142
LHS	**L**eft **H**and **S**ide.	p. 41
MEMS	**M**icro **E**lectro-**M**echanical **S**ystems.	p. 44
Modelica	An object-oriented software language for the modelling of physical systems.	p. 53
MOSA	**M**ulti-**O**bjective ***S**imulated **A**nnealing* (Suppapitnarm et al. 1999).	p. 43
MTM	**M**otion **T**ransformation **M**atrix. Used by Kota and Chiou (1992) to represent rotation and translation.	p. 37

Objective Function	A quantifiable measure of 'goodness' of a design for use with search algorithms.	p. 54
Optimally Directed Design (entity)	A design in the co-ordinate range of the global optimum.	p. 43
Optimally Directed Design (process)	'Design optimisation that directs […] design generation towards the numeric range of a global optimum' (Shea 1997). See section 3.6.	p. 55
Parallel Grammar	A grammar for generating representations of form and function.	p. 66
Pareto Archive	Set of non-dominated solutions to an optimisation problem. See section 6.4.2.	p. 145
Pattern Search	A deterministic search algorithm that seeks to repeat successful sequences of design modifications to increase search efficiency.	p. 56
PFRS	**P**roduct **F**amily **R**easoning **S**ystem (Siddique and Rosen 2001).	p. 46
Pitch Radius	The pitch radii of two meshing spur gears describe the circles that remain tangent throughout the engagement cycle (Marghitu et al. 2001).	p. 75
Production System	A formalism for generation. See section 3.3 and (Gips and Stiny 1980).	p. 37
P-Rule	**P**erturb **R**ule. A grammar rule that modifies, i.e. perturbs, a structure. See also *C-Rule*.	p. 103
RCD	**R**ing-plate **C**ycloid **D**rive. A type of *EGT* mechanism.	p. 46
RD	**R**andom **D**ownhill search. A random search method that accepts moves if an objective function improvement is recorded. Not to be confused with hill climbing, which is a gradient-based method.	p. 112
RHS	**R**ight **H**and **S**ide.	p. 42
SA	**S**imulated **A**nnealing is a stochastic search method that is analogous to 'annealing', a heat treatment process for metal alloys.	p. 57
Satisfice	To 'decide on and pursue a course of action that will satisfy the minimum requirements necessary to achieve a particular goal' (source: Oxford English Dictionary).	p. 41

Skeleton Model	A geometry-based design *heuristic* used by Wahl et al. (2003) to assess the 'goodness' of mechanisms.	p. 54
SLS	**S**elective **L**aser **S**intering. A rapid prototyping technique, products of which are referred to as 'SLS models'.	p. 171
Transmission	In mechanical engineering, the components that transmit power from the driveshaft to the point of power application, i.e. in the case of automotive engineering the driven wheels of a vehicle.	p. 20
VLSI	**V**ery **L**arge **S**cale **I**ntegration.	p. 35
VRML	**V**irtual **R**eality **M**odelling **L**anguage.	p. 88

10. References

Adams, D (1988), *The Hitchhiker's Guide To The Galaxy*, Tor Books.

Agarwal, M (1999), *Supporting Automated Design Generation: Function Based Shape Grammars and Insightful Optimization*, PhD Thesis, Carnegie Mellon University, Pittsburgh.

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

Alber, R and Rudolph, S (2003), "'43' - A Generic Approach for Engineering Design Grammars", *AAAI'03 Spring Symposium: Computational Synthesis*, Stanford, California, pp. 11-17.

Alber, R, Rudolph, S and Kröplin, B (2002), "On Formal Languages in Design Generation and Evolution", *WCCM V: Fifth World Congress on Computational Mechanics*, Vienna, Austria.

Al-Hakim, L, Kusiak, A and Mathew, J (2000), "A graph-theoretic approach to conceptual design with functional perspectives", *Computer-Aided Design*, **32**(14), pp. 867-875.

Andersson, J (2001), *Multiobjective Optimization in Engineering Design*, PhD thesis, Linköpings University, Linköpings.

Antonsson, EK (1997), "The Potential for Mechanical Design Compilation", *Research in Engineering Design*, **9**(4), pp. 191-194.

Antonsson, EK and Cagan, J (2001), *Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge.

Antonsson, EK and Whitney, DE (1997), "Correspondence", *Research in Engineering Design*, **9**(4), pp. 246-247.

Arzethauser, K (2003), "Gears galore", in *Watch International*, G. Kern and H. A. Pantli, eds., **13**(3), pp. 56-59.

Ashby, M and Johnson, K (2002), *Materials and Design*, Butterworth-Heinemann, Oxford.

Ashby, MF (1992), *Materials Selection in Mechanical Design*, Butterworth-Heinemann, Oxford.

Ashby, MF and Jones, DRH (1998), *Engineering Materials 2*, Butterworth-Heinemann, Oxford.

Bamsey, I, Smith, C, Staniforth, A and McDermott, M (2001), "Transmission Symphony: Audi R8 Transmission by Ricardo", *Race Tech*, **6**(5), pp. 33-40.

Blessing, LTM, Chakrabarti, A and Wallace, KM (1998), "An Overview of Descriptive Studies in Relation to a General Design Research Methodology", in *The Key to Successful Product Development*, E. Frankenberger, P. Badke-Shaub, and H. Birkhofer, eds., Springer-Verlag, pp. 42-56.

Bolognini, F (2003), *The Performance Aspects of Mechanical Design Synthesis*, CUED/C-EDC/TR125, Department of Engineering, Cambridge University, Cambridge.

Boothroyd, G and Dewhurst, P (1989), *Product Design for Assembly*, Boothroyd Dewhurst, Inc., Wakefield, RI.

Boothroyd, G, Dewhurst, P and Knight, W (1994), *Product Design for Manufacture and Assembly*, Marcel Dekker, Inc, New York.

Bracewell, RH (2002), "Synthesis based on function-means trees: Schemebuilder", in *Engineering Design Synthesis*, A. Chakrabarti, ed., Springer Verlag, London, pp. 199-212.

Bracewell, RH and Johnson, AL (1999), "From embodiment generation to virtual prototyping", *International Conference on Engineering Design, ICED 99*, Munich, pp. 685-690.

Bracewell, RH and Sharpe, JEE (1994), "The use of Bond Graph Methodology in an Integrated Interdisciplinary Design System", *Joint Hungarian-British Mechatronics Conference*, Budapest, pp. 595-600.

Bracewell, RH and Sharpe, JEE (1996), "Functional descriptions used in computer support for qualitative scheme generation - "Schemebuilder"", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*(10), pp. 333-346.

Bracewell, RH and Shea, K (2001), "CAEDRE: A Product Platform to Support Creation and Evaluation of Advanced Computer Aided Engineering Tools", *13th International Conference on Engineering Design*, Glasgow, UK, pp. 539-546.

Bracewell, RH, Shea, K, Langdon, PM, Blessing, LS and Clarkson, PJ (2001), "A Methodology for Computational Design Tool Research", *13th International Conference on Engineering Design*, Glasgow, UK, pp. 181-188.

Broenink, JF (1999), *Introduction to Physical Systems Modelling with Bond Graphs*, University of Twente, Dept EE, Control Laboratory.

Brown, K (1997), "Grammatical Design", *IEEE Expert*, **12**(2), pp. 27-33.

Brown, KN, McMahon, CA and Sims Williams, JH (1995), "Features, aka The Semantics of a Formal Language of Manufacturing", *Research in Engineering Design*, **7**, pp. 151-172.

Burgess, SC, Moore, DF, Newland, DE and Klaubert, HL (1997), "A Study of Mechanical Configuration Optimisation in Micro-systems", *Research in Engineering Design*, **9**(1), pp. 46-60.

Cagan, J (1990), *Innovative Design of Mechanical Structures from First Principles*, PhD Thesis, University of California at Berkeley, California.

Cagan, J (1994), "Shape annealing solution to the constrained geometric knapsack problem", *Computer-Aided Design*, **26**(10), pp. 763-770.

Cagan, J and Agogino, AM (1987), "Innovative Design of Mechanical Structures from First Principles", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **1**(3), pp. 169-189.

Cagan, J and Mitchell, WJ (1993), "Optimally directed shape generation by shape annealing", *Environment and Planning B: Planning and Design*, **20**, pp. 5-12.

Cagan, J, Szykman, S, Clark, R, Dastidar, P and Weisser, P (1996), "HVAC CAD layout tools: a case study of university/industrial collaboration", *Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine, California.

Cagan, J and Vogel, CM (2002), *Creating Breakthrough Products*, Prentice Hall PTR, Upper Saddle River.

Callahan, S and Heisserman, J (1997), "A Product Representation to Support Process Automation", in *Product Modeling for Computer Integrated Design and Manufacture*, M. Pratt, R. Sriram, and M. Wozny, eds., Chapman and Hall, London, pp. 285-296.

Campbell, MI, Cagan, J and Kotovsky, K (1999), "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment", *Research in Engineering Design*, **11**, pp. 172-192.

Campbell, MI, Cagan, J and Kotovsky, K (2000), "Agent-Based Synthesis of Electromechanical Design Configurations", *Journal of Mechanical Design*, **122**(1), pp. 61-69.

Campbell, MI, Cagan, J and Kotovsky, K (2003), "The A-Design Approach to Managing Automated Design Synthesis", *Research in Engineering Design*, **14**(1), pp. 12-24.

Campbell, MI and Hasad, A (2003), "Design Evaluation Method for the Disassembly of Electronic Equipment", *International Conference on Engineering Design, ICED'03*, Stockholm, Sweden.

Campbell, MI and Rai, R (2003), "A Generalization of Computational Synthesis Methods in Engineering Design", *AAAI'03 Spring Symposium: Computational Synthesis*, Stanford, CA, pp. 34-41.

Chakrabarti, A and Bligh, TP (1996), "An approach to functional synthesis of mechanical design concepts: theory, applications and emerging research issues", *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, **10**(4), pp. 313-331.

Chakrabarti, A and Bligh, TP (2001), "A scheme for functional reasoning in conceptual design", *Design Studies*, **22**, pp. 493-517.

Chakrabarti, A and Johnson, A (1999), "Detecting Side Effects in Solution Principles", *International Conference on Engineering Design, ICED 99*, Munich, pp. 661-666.

Chakrabarti, A, Langdon, P, Liu, Y-C and Bligh, TP (2002), "An approach to compositional synthesis of mechanical design concepts using computers", in *Engineering Design Synthesis*, A. Chakrabarti, ed., Springer Verlag, London, pp. 179-197.

Chandrasekaran, B and Josephson, JR (2000), "Function in Device Representation", *Engineering with Computers, special Issue on Computer Aided Engineering*, **16**, pp. 162-177.

Chase, SC (1996), "Design Modeling With Shape Algebras and Formal Logic", *ACADIA '96*, Tucxon, AZ.

Chase, SC (2002), "(Re)design of construction assemblies with function/behaviour/structure grammars", *Design e-ducation: Connecting the Real and the Virtual, Proceedings of the 20th Conference on Education in Computer Aided Architectural Design in Europe*, Warsaw, pp. 356-359.

Chase, SC and Liew, P (2001), "A Framework for Redesign using FBS Models and Grammar Adaptation", *CAAD Futures 2001*, Eindhoven, The Netherlands.

Chiou, S-J and Kota, S (1999), "Automated conceptual design of mechanisms", *Mechanism and Machine Theory*, **34**, pp. 467-495.

Chomsky, N (1957), *Syntactic Structures*, Mouton, The Hague.

Couper, AS (1858), "On a new chemical theory", *The Philosophical Magazine and Journal of Science*, **16**, pp. 104-116.

De Kleer, J and Brown, JS (1984), "A Qualitative Physics Based on Confluences", *Artificial Intelligence*, **24**, pp. 7-83.

Deb, K and Jain, S (2003), "Multi-Speed Gearbox Design Using Multi-Objective Evolutionary Algorithms", *Journal of Mechanical Design*, **125**(3), pp. 609-619.

Duarte, JP (2003), "A Discursive Grammar for Customizing Mass Housing", *Digital Design: 21st International eCAADe Conference*, Graz University of Technology, Austria, pp. 665-674.

Duarte, JP (in progress), "Customizing Mass Housing: The grammar of Alvaro Siza's Houses at Malagueira", *Environment and Planning B*.

Dym, CL (1994), *Engineering Design: A Synthesis of Views*, Cambridge University Press.

Dym, CL and Levitt, RE (1991), *Knowledge-Based Systems in Engineering*, McGraw-Hill Inc, New York.

Earl, CF (1987), "Shape grammars and the generation of designs", in *Principles of Computer-aided Design*, J. Rooney and J. P. Steadman, eds., Pitman Publishing, London, pp. 297-315.

Egginger, U (2003), "Eternity in an Hour", in *Watch International*, G. Kern and H. A. Pantli, eds., **13**(1), pp. 39-41.

Elmo, Gum, Heather, Holly, Mistletoe and Rowan (2002), *Notes towards the Complete Works of Shakespeare*, Paignton Zoo Environmental Park.

Finger, S and Rinderle, JR (1989), "A Transformational Approach to Mechanical Design using a Bond Graph Grammar", *First ASME Design Theory and Methodology Conference*, Montreal, Quebec, pp. 107-116.

Finger, S and Rinderle, JR (2002), "Transforming behavioural and physical representations of mechanical designs", in *Engineering Design Synthesis*, A. Chakrabarti, ed., Springer Verlag, London, pp. 303-317.

Forbus, KD (1984), "Qualitative process theory", *Artificial Intelligence*, **24**, pp. 85-168.

Forbus, KD, Nielsen, P and Faltings, B (1991), "Qualitative spatial reasoning: The clock project", *Artificial Intelligence*, **51**, pp. 417-471.

French, M (1994), *Invention and Evolution*, Second ed., Cambridge University Press.

French, M (1999), *Conceptual Design for Engineers*, Third ed., Springer-Verlag, London.

Fricke, G (1996), "Successful Individual Approaches in Engineering Design", *Research in Engineering Design*, **8**(3), pp. 151-165.

Gips, J and Stiny, G (1980), "Production Systems and Grammars: a Uniform Characterization", *Environment and Planning B*, **7**, pp. 399-408.

Glover, F and Laguna, M (1997), *Tabu Search*, Kluwer Academic Publishers, Boston, MA.

Gustafson, GB and Wilcox, CH (1998), *Analytical and Computational Methods of Advanced Engineering Mathematics*, Springer-Verlag, New York.

Hansen, CT and Andreasen, MM (2002), "Two approaches to synthesis based on the domain theory", in *Engineering Design Synthesis*, A. Chakrabarti, ed., Springer Verlag, London, pp. 93-108.

Heisserman, J and Mattikalli, R (1998), "Representing Relationships in Hierarchical Assemblies", *ASME Design Engineering Technical Conferences*, Atlanta, Georgia, USA.

Heisserman, J and Woodbury, R (1994), "Geometric design with boundary solid grammars", in *Formal Design Methods for CAD*, J. Gero and E. Tyugu, eds., Elsevier Science BV, North-Holland, Amsterdam, pp. 85-105.

Hewitt, K (2003), *Black and Decker*, Private Communication, 2 April 2003

Holland, JH (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

Hooke, R and Jeeves, TA (1961), "Direct Search Solution of Numerical and Statistical Problems", *Journal of the Association for Computing Machinery (ACM)*, **8**(2), pp. 212-229.

Hoover, SP and Rinderle, JR (1989), "A Synthesis Strategy for Mechanical Devices", *Research in Engineering Design*, **1**, pp. 87-103.

James, B (2003), *Romax Technology*, Private Communication, 7 November 2003

Jensen, T (2000), "Function Integration Explained by Allocation and Activation of Wirk Elements", *ASME 2000 Design Engineering Technical Conferences*, Baltimore, Maryland.

Johnson, S (2001), *Emergence*, Penguin, London.

Karnopp, D and Rosenberg, RC (1968), *Analysis and Simulation of Multiport Systems: The Bond Graph Approach to Physical System Dynamics*, The MIT Press, Cambridge, Massachusetts.

Karnopp, DC, Margolis, DL and Rosenberg, RC (2000), *System Dynamics*, Third ed., John Wiley & Sons, Inc, New York.

Kirkpatrick, S, Gelatt Jr, C and Vecchi, M (1983), "Optimization by Simulated Annealing", *Science*, **220**(4598), pp. 671-680.

Knight, TW (1994), *Transformations in Design*, Cambridge University Press, Cambridge.

Kota, S and Chiou, S-J (1992), "Conceptual Design of Mechanisms Based on Computational Synthesis and Simulation of Kinematic Building Blocks", *Research in Engineering Design*, **4**, pp. 75-87.

Koza, JR, Keane, MA and Streeter, MJ (2003), "What's AI Done for Me lately? Genetic Programming's Human-Competitive Results", *IEEE Intelligent Systems*, **18**(3), pp. 25-31.

Kron, G (1963), *Diakoptics: the piecewise solution of large-scale systems*, MacDonald, London.

Krstic, D (2001), "Algebras and grammars for shapes and their boundaries", *Environment and Planning B: Planning and Design*, **28**, pp. 151-162.

Lam, J and Delosme, J-M (1998), "Performance of a New Annealing Schedule", *Proceedings of the 15th ACM/IEEE Design Automation Conference*, Anaheim, CA, US, pp. 306-311.

Li, X, Schmidt, L, He, W, Li, L and Qian, Y (2001), "Transformation of an EGT Grammar: New Grammar, New Designs", *ASME 2001 Design Engineering Technical Conferences*, Pittsburgh, Pennsylvania.

Li, X and Schmidt, LC (2000), "Grammar-Based Designer Assistance Tool for Epicyclic Gear Trains", *ASME 2000 Design Engineering Technical Conferences*, Baltimore, Maryland.

Liew, P and Chase, SC (2001), "Describing Designs with Function-Behaviour-Structure Graphs", *Mathematics and Design*, Geelong and Melbourne, Australia.

Lipson, H and Pollack, JB (2000a), "Evolution of Physical Machines: Towards Escape Velocity", *6th International Conference on Artificial Intelligence in Design, AID'00*, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, pp. 269-285.

Lipson, H and Pollack, RB (2000b), "Automatic Design and Manufacture of Robotic Lifeforms", *Nature*, **406**(6799), pp. 974-978.

Liu, YC, Chakrabarti, A and Bligh, TP (1999), "Transforming Functional Solutions into Physical Solutions", *ASME Design Theory and Methodology Conference*, Las Vegas, Nevada.

Liu, Y-C, Bligh, T and Chakrabarti, A (2003), "Towards an 'ideal' approach for concept generation", *Design Studies*, **24**, pp. 341-355.

Marghitu, DB, Diaconescu, CI and Craciunoiu, N (2001), "Machine Components", in *Mechanical Engineer's Handbook*, D. B. Marghitu, ed., Academic Press, San Diego, California, pp. 244-339.

Martello, S and Toth, P (1990), *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, Ltd, New York.

McCormack, J and Cagan, J (2000), "Enabling the use of Shape Grammars: Shape Grammar interpretation through general shape recognition", *ASME Design Engineering Technical Conference*, Baltimore, Maryland.

McCormack, JP and Cagan, J (2003), "Increasing the Scope of Implemented Shape Grammars: A Shape Grammar Interpreter for Curved Shapes", *ASME 2003 International Design Engineering Technical Conferences*, Chicago, Illinois, USA.

McCormack, JP, Cagan, J and Vogel, CM (2002), "Crossing the '63 Riviera with a Concept Cielo: Capturing, Representing and Generating the Buick Brand", *ASME 2002 Design Engineering Technical Conferences*, Montreal, Canada.

Moon, Y-M and Kota, S (2002), "Automated synthesis of mechanisms using dual-vector algebra", *Mechanism and Machine Theory*, **37**, pp. 143-166.

Newell, A and Simon, HA (1972), *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

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

Ousterhout, JK (1998), "Scripting: Higher Level Programming for the 21st Century", *IEEE Computer*, **31**(3), pp. 23-30.

Pahl, G and Beitz, W (1996), *Engineering Design*, Second ed., Springer.

Pahl, G and Wallace, K (2002), "Using the Concept of Functions to help Synthesis Solutions", in *Engineering Design Synthesis*, A. Chakrabarti, ed., Springer-Verlag, London, pp. 109-119.

Papalambros, PY and Wilde, DJ (2000), *Principles of Optimal Design*, Cambridge University Press.

Pareto, V (1906), *Manual of Political Economy*, 1971 translation of 1927 ed., A. M. Kelley, New York.

Paynter, HM (1961), *Analysis and Design of Engineering Systems*, MIT Press, Cambridge, USA.

Petrovski, H (1994), *Design Paradigms: Case Histories of Error and Judgement in Engineering*, Cambridge University Press, Cambridge, UK.

Poon, SY (2003), *Romax Technology*, Private Communication, 21 November 2003

Popplestone (1987), "The Edinburgh Design Systems as a Framework for Robotics: The Design of Behavior", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **1**(1), pp. 25-36.

Post, E (1943), "Formal reductions of the general combinatorial decision problems", *American Journal of Mathematics*, **65**, pp. 197-268.

Pugliese, M and Cagan, J (2002), "Capturing a Rebel: Modeling the Harley-Davidson Brand through a Motercycle Shape Grammar", *Research in Engineering Design*, **13**(3), pp. 139-156.

Raphael, B and Smith, IFC (2003), *Computer-Aided Engineering*, John Wiley & Sons Ltd, Chichester.

Reeves, CR (1996), "Modern Heuristic Techniques", in *Modern Heuristic Search Methods*, V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, eds., John Wiley & Sons Ltd, Chichester, pp. 1-25.

Rosen, DW (1993), "Feature-Based Design: Four Hypotheses for Future CAD Systems", *Research in Engineering Design*, **5**(3), pp. 125-139.

Schmidt, LC and Cagan, J (1995), "Recursive Annealing: A Computational Model for Machine Design", *Research in Engineering Design*, **7**, pp. 102-125.

Schmidt, LC and Cagan, J (1997), "GGREADA: A Graph Grammar-based Machine Design Algorithm", *Research in Engineering Design*, **9**, pp. 195-213.

Schmidt, LC and Cagan, J (1998), "Optimal Configuration Design: An Integrated Approach using Grammars", *Journal of Mechanical Design*, **120**(9), pp. 2-9.

Schmidt, LC, Shetty, H and Chase, SC (2000), "A Graph Grammar Approach for Structure Synthesis of Mechanisms", *Journal of Mechanical Design*, **122**, pp. 371-376.

Schmidt, LC, Shi, H and Kerkar, S (1999), "The "Generation Gap": A CSP Approach Linking Function to Form Grammar Generation", *ASME Engineering Technical Conference*, Las Vegas, Nevada.

Sharpe, JEE (1978), "Application of Bond Graphs to the Synthesis and Analysis of Telechirics and Robots", *3rd Symposium on Theory and Practice of Robots and Manipulators, Third International CISM-IFToMM Symposium*, Udine, Italy, pp. 217-227.

Shea, K (1997), *Essays of Discrete Structures: Purposeful Design of Grammatical Structures by Directed Stochastic Search*, PhD Thesis, Carnegie Mellon University, Pittsburgh.

Shea, K, Aish, R and Gourtovaia, M (2003), "Towards Integrated Performance-Based Generative Design Tools", *Digital Design: 21st International eCAADe Conference*, Graz University of Technology, Austria, pp. 553-560.

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

Shea, K and Cagan, J (1998), "The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent", *Design Studies*, **20**, pp. 3-23.

Shea, K and Smith, IFC (1999), "Applying shape annealing to full-scale transmission tower re-design", *Proceedings of DETC99: 1999 ASME Design Engineering Technical Conferences*, Las Vegas, NV.

Shea, K and Starling, AC (2003), "From Discrete Structures to Mechanical Systems: A Framework for Creating Performance-Based Parametric Synthesis Tools", *AAAI'03 Spring Symposium: Computational Synthesis*, Stanford, CA, pp. 210-217.

Shi, H (2003), "Partitioning Knowledge for Generative Configuration Design", *ASME 2003 Design Engineering Technical Conferences*, Chicago, Illinois, USA.

Siddique, Z and Rosen, DW (1999), "Product Platform Design: A Graph Grammar Approach", *ASME Design Engineering Technical Conference*, Las Vegas, Nevada.

Siddique, Z and Rosen, DW (2001), "On combinatorial design spaces for the configuration design of product families", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**(2), pp. 91-108.

Simon, HA (1973), "The Structure of Ill-Structured Problems", *Artificial Intelligence*, **4**(3), pp. 181-201.

Simon, HA (2001), *The Sciences of the Artificial*, Third ed., The MIT Press, Cambridge, Massachusetts.

Sobel, D (1998), *Longitude*, Fourth Estate, London.

Sriram, RD (1997), *Intelligent Systems for Engineering: A Knowledge-Based Approach*, Springer-Verlag, London.

Stahovich, TF (2001), "Artificial Intelligence for Design", in *Formal Engineering Design Synthesis, Chapter 7*, E. K. Antonsson and J. Cagan, eds., Cambridge University Press, Cambridge, pp. 228-269.

Stahovich, TF and Kara, LB (2001), "A representation for comparing simulations and computing the purpose of geometric features", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**(2), pp. 189-201.

Standish, TA (1998), *Data Structures in Java*, Addison-Wesley.

Starling, AC and Shea, K (2002), "A Clock Grammar: The Use of a Parallel Grammar in Performance-Based Mechanical Design Synthesis", *2002 ASME International Design Engineering Technical Conferences*, Montreal, Canada.

Starling, AC and Shea, K (2003), "A Grammatical Approach to Computational Generation of Mechanical Clock Designs", *International Conference on Engineering Design, ICED'03*, Stockholm, Sweden.

Steinmetz (1909), *Theory and Calculation of Transient Electric Phenomena and Oscillations*, McGraw Publishing Co., New York.

Stiny, G (1980a), "Introduction to Shape and Shape Grammars", *Environment and Planning B*, **7**, pp. 343-351.

Stiny, G (1980b), "Kindergarten Grammars: designing with Froebel's building gifts", *Environment and Planning B*, **7**, pp. 409-462.

Stiny, G (1990), "What is a design?", *Environment and Planning B: Planning and Design*, **17**, pp. 97-103.

Stiny, G (1991), "The Algebras of Design", *Research in Engineering Design*, **2**, pp. 171-181.

Stiny, G (1992), "Weights", *Environment and Planning B: Planning and Design*, **19**, pp. 413-430.

Stone, RB and Wood, KL (2000), "Development of a Functional Basis for Design", *Journal of Mechanical Design*, **122**, pp. 359-370.

Suh, NP (2001), *Axiomatic Design*, Oxford University Press, New York.

Suppapitnarm, A, Seffen, KA, Parks, GT, Clarkson, PJ and Liu, JS (1999), "Design by Multiobjective Optimisation using Simulated Annealing", *ICED'99: International Conference on Engineering Design*, Munich.

Szykman, S and Cagan, J (1995), "A Simulated Annealing-Based Approach to Three-Dimensional Component Packing", *Journal of Mechanical Design*, **117**(June), pp. 308-314.

Szykman, S and Cagan, J (1997), "Constrained Three-Dimensional Component Layout Using Simulated Annealing", *Journal of Mechanical Design*, **119**(1), pp. 28-35.

Szykman, S, Schmidt, LC and Shetty, H (1997), "Improving the Efficiency of Simulated Annealing Optimization through Detection of Productive Search", *ASME Design Engineering Technical Conferences*, Sacramento, California, US.

Tay, EH, Flowers, W and Barrus, J (1998), "Automated Generation and Analysis of Dynamic System Designs", *Research in Engineering Design*, **10**, pp. 15-29.

Thoma, JU (1975), *Introduction to Bond Graphs and their Applications*, Pergamon Press, Oxford.

Thornton, AC (1993), *Constraint Specification and Satisfaction in Embodiment Design*, PhD thesis, University of Cambridge, Cambridge.

Tiller, M (2001), *Introduction to Physical Modeling with Modelica*, Kluwer Academic Publishers, Boston.

Torczon, V (1997), "On the Convergence of Pattern Search Algorithms", *SIAM Journal on Optimization*, **7**(1), pp. 1-25.

Ulrich, KT and Eppinger, SD (1995), *Product Design and Development*, McGraw-Hill.

Ulrich, KT and Seering, WP (1987), "Achieving Multiple Goals in Conceptual Design", *Intelligent CAD, I. Proceedings of IFIP TC/WG 5.2 Workshop on Intelligent CAD*, Boston, USA, pp. 213-222.

Ulrich, KT and Seering, WP (1989), "Synthesis of Schematic Descriptions in Mechanical Design", *Research in Engineering Design*, **1**(1), pp. 3-18.

Ulrich, KT and Seering, WP (1990), "Function Sharing in Mechanical Design", *Design Studies*, **11**(4), pp. 223-234.

Umeda, Y, Tadeda, T, Tomiyama, T and Yoshikawa, H (1990), "Function, Behaviour, and Structure", *Applications of Artificial Intelligence in Engineering V: Design*, Berlin, pp. 177-193.

Vale, CAW and Shea, K (2003), "Learning Intelligent Modification Strategies in Design Synthesis", *AAAI'03 Spring Symposium 'Computational Synthesis'*, Stanford, CA.

Wahl, J-C, Sartor, M and Fauroux, J-C (2001), "Using the Skeleton Model for Preliminary Geometrical Synthesis of 3D Kinematic Chains", *ICED'01: 13th International Conference on Engineering Design*, Glasgow, UK.

Wahl, J-C, Sartor, M and Paredes, M (2003), "A General Framework for Automated Conceptual Design of One-DOF Mechanisms", *International Conference on Engineering Design, ICED'03*, Sweden, Stockholm.

Ward, AC (2001), "Mechanical Design Compilers", in *Formal Engineering Design Synthesis, chapter 12*, E. K. Antonsson and J. Cagan, eds., Cambridge University Press, Cambridge, pp. 428-441.

Welch, RV and Dixon, JR (1991), "Conceptual Design of Mechanical Systems", *Proceedings of the 1991 ASME Design Theory and Methodology Conference*, Miami, Florida, pp. 61-68.

Whitney, D (1996), "Why mechanical design cannot be like VLSI design", *Research in Engineering Design*, **8**(3), pp. 124-138.

Wielinga, B and Schreiber, G (1997), "Configuration-Design Problem Solving", *IEEE Expert*, **12**(2), pp. 49-56.

Winsor, J and MacCallum, K (1994), "A Review of Functionality Modelling in Design", *The Knowledge Engineering Review*, **9**(2), pp. 163-199.

Winston, PH (1993), *Artificial Intelligence*, Third ed., Addison Wesley.

Yan, H-S (1998), *Creative Design of Mechanical Devices*, Springer-Verlag, Singapore.

Yin, S and Cagan, J (1998), "A Pattern Search-Based Algorithm for Three-Dimensional Component Layout", *Proceedings of the 24th ASME Design Automation Conference (DAC-5582)*, Atlanta, Georgia, USA.

Yin, S and Cagan, J (2000a), "Exploring the Effectiveness of Various Patterns in an Extended Pattern Search Layout Algorithm", *ASME Design Engineering Technical Conferences*, Baltimore, MD, USA.

Yin, S and Cagan, J (2000b), "An Extended Pattern Search Algorithm for Three-Dimensional Component Layout", *Journal of Mechanical Design*, **122**(March), pp. 102-108.

Yin, S, Cagan, J, Hodges, P and Li, X (1999), "Layout of an Automobile Transmission using Three-Dimensional Shapeable Components", *Proceedings of the 25th ASME Design Automation Conference (DAC-8564)*, Las Vegas, Nevada, USA.