AN ABSTRACTION-BASED METHODOLOGY FOR MECHANICAL CONFIGURATION DESIGN

by

Gary Lee Snavely

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
1992

CHAPTER 1

INTRODUCTION

Among the numerous ways of decomposing the mechanical design process, the decomposition suggested by Ullman [1992] is original design, parametric design, configuration design and catalog selection design. Original design involves the development of a process, component, or assembly not previously in existence. Designs of this type are extremely difficult to generate automatically as is evidenced by the lack of attempts at automatic generation [Snavely et. al. 1990]. Parametric design involves assigning values to a set of design variables. The automation of parametric design is extremely well researched, with optimization methods gaining distinction as perhaps the only formalized mechanical design methodology [Papadimitriou and Steiglitz 1982, and Papalambros and Wilde 1988]. The other two types of design, configuration and catalog selection, fall somewhere between original and parametric design. Configuration design and catalog selection design are the focus of this thesis.

1.1 Configuration Design

Mittal and Frayman [1989] define configuration design as a special type of design activity in which the artifact being designed is assembled from a set of pre-defined components that can only be connected together in certain ways. The task of a computational methodology for performing configuration design can be described as follows:

Given:
- a fixed, pre-defined set of components,
- some description of the desired configuration,
- possibly some criteria for making optimal selections.

Build:
- a (possibly empty) set of configurations that satisfy all requirements.

Where:
- a component is described by a set of properties, ports for connection to other components, and constraints at each port that describe the components that can be connected at that port,
- a configuration is a set of components and a description of the connections between components in the set.

As one might suspect, the computational size of the configuration problem is large. In fact, the general configuration problem is infinitely large. One way to place a finite limit on the configuration problem is to introduce a finite upper bound on the number of components in a configuration. Even with such an upper bound, Mittal and Frayman [1989] have shown that the number of configurations that can be generated for a given configuration problem is exponential in that upper bound. Thus, any computational method of performing configuration design must include some means of effectively reducing the size of the problem.

1.2 Catalog Selection Design

Catalog selection is a process of replacing each component in a fixed configuration with a more specific instance, or offspring, of that component. The task of a computational methodology for performing catalog selection design can be described as follows:

Given:
- a fixed configuration of components,
- sets of offspring components,
- some description of the desired configuration,
- possibly some criteria for making optimal selections.

Build:
- a (possibly empty) set of configurations that satisfy all requirements.

Where:
- a configuration is a set of components and a description of the connections between the components in the set,
- each component in the configuration has associated with it a set of offspring components.

The catalog selection problem is similar to the configuration problem. In fact, the catalog selection problem is also exponential in the number of components in the configuration [Ward 1989]. However, a major difference between configuration and catalog selection is that the fixed configurations in the catalog selection problem are of finite length, thereby limiting the catalog selection problem to a finite number of combinations.

1.3 Abstraction-Based Configuration Design

Both configuration and catalog selection are combinatorial problems. Some have suggested that one way of reducing the size of a combinatorial problem is to solve the problem at a higher level of abstraction, in which less important details are temporarily ignored [Sacerdoti 1974, Lee 1992]. Solutions that appear promising at the higher level of abstraction are then made more specific by recursively introducing finer levels of detail.

The **A**bstraction-**B**ased **CON**figuration (ABCON) design methodology presented in this thesis incorporates one means of reducing the size of the configuration problem by abstracting components to higher levels of abstraction. ABCON abstracts user-defined components to a higher abstraction level, using a highly structured, domain-independent procedure. At higher abstraction levels, less important detail is temporarily ignored, and each component represents a family of lower-level components. Configuration design is then performed at this high level, explicitly enumerating all configurations at that level. Any complete configuration at the highest level is instantiated to lower levels by recursively performing the catalog selection process. At the same time, any incomplete configuration at the highest level is eliminated, thereby eliminating all possible lower-level instantiations of that configuration. In this manner, all configurations of components at the lowest, user-defined level of abstraction are implicitly enumerated.

A simple example will illustrate the concepts of abstraction, configuration and instantiation, as well as give some insight into the manner in which ABCON reduces the size of the explicit enumeration problem. Consider the database of six components shown in Figure 1.1. The first two components, labeled S_1 and S_2, represent sources of electricity. The next two components, labeled T_1 and T_2, represent electrical transformers. The last two components, labeled M_1 and M_2, represent electrical motors. Each component has a number of ports by which it can be connected to other components. Each port, represented by a black dot, is labeled with a set of specifications that describe the interaction that takes place between components at that port. The first specification in each set specifies alternating current (ac) or direct current (dc), while the second specification in each set specifies the number of volts. Components can be connected to form configurations of components by connecting ports that have identical port specifications.

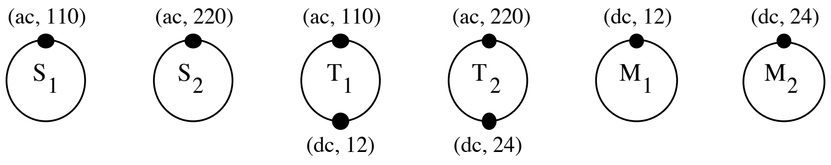

Figure 1.1 Component Database

One simple configuration problem in this domain is to connect the source of 110 volt ac power, component S_1, to the 12 volt dc motor, component M_1, in all feasible ways. In order to keep this problem simple, each component in the database may be used no more than once in any configuration.

Figure 1.2 shows a search tree representing the steps an explicit enumeration methodology would take to solve this problem. The components S_1 and M_1 are called required components and are represented as squares in this figure to denote that they must be included in any feasible solution. All other components, called optional components, are represented as circles.

At the first branch in the search tree, the specifications of every port of every component in the database are compared to the specifications of the single port of component S_1 to determine if there is a match. In the figure, ports that match are connected by a solid line, ports that do not match are connected by a dashed line, and ports that have not yet been compared are represented by a black dot. In this instance, the only port found to match the single port of S_1 is the 110 volt ac port of component T_1. Therefore, the configuration that includes S_1, T_1, and the line connecting the two, is the only feasible configuration resulting from the first step in the explicit enumeration process.

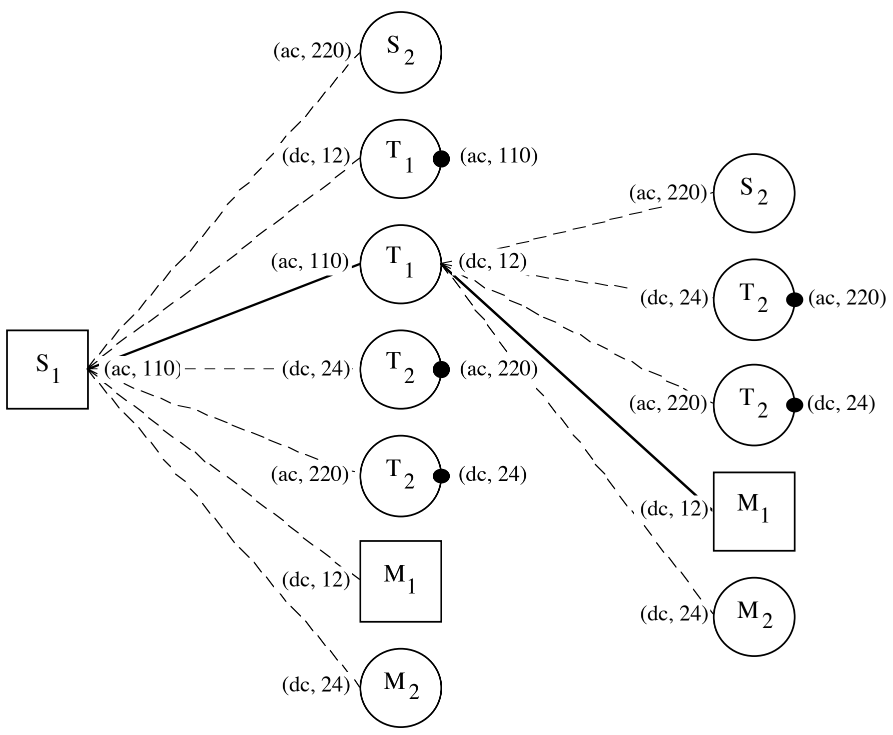

Figure 1.2 Explicit Enumeration Configuration

At the next branch of the search tree, all remaining ports are compared to the 12 volt dc port of component T_1 to determine if there is a match. Once again, only a single port matches: the 12 volt dc port of component M_1. The explicit enumeration process has identified one feasible solution that satisfies the requirement that components S_1 and M_1 be connected. Indicative of the amount of computation required to arrive at this single solution, the explicit enumeration process expanded nodes of the search tree along 12 different branches.

Now consider a slight modification to the original database. Figure 1.3 represents a hierarchical database in which each of the original six components has been abstracted to a higher level of abstraction. The process of abstracting a component is simply a process of eliminating the last specification on each component port. Elimination of the last specification, the number of volts, causes S_1 and S_2 to abstract to the same component, T_1

and T_2 to abstract to the same component, and M_1 and M_2 to abstract to the same component.

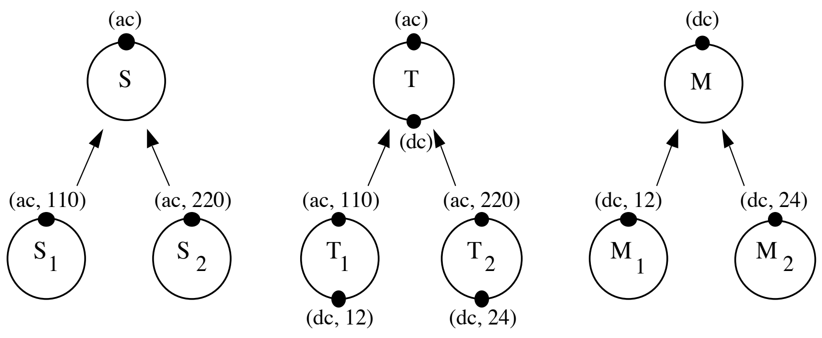

Figure 1.3 ABCON Hierarchical Database

The same configuration problem that was explicitly solved above will now be solved using the ABCON methodology. The first step in the ABCON methodology, called topology generation, is to explicitly perform the configuration process at the highest level of abstraction. Figure 1.4 shows a search tree representing the steps the topology generation methodology would take to explicitly generate all feasible topologies at the highest level. Once again, the required components, or more accurately, the abstracted parents of the required components, are represented as squares to denote that they must be included in all feasible configurations, and optional components are represented as circles. The explicit enumeration of all feasible solutions at the abstracted level has located a single feasible solution as indicated by the solid lines between components in Figure 1.4.

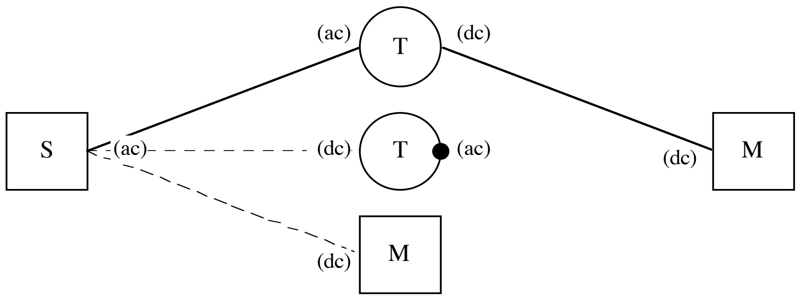

Figure 1.4 Topology Generation

The next step in the ABCON methodology, called topology instantiation, is to recursively instantiate all feasible solutions to lower levels of abstraction using the catalog selection process. Figure 1.5 represents the instantiation of the single feasible solution found in the topology generation step. In the topology instantiation process, each of the required components in a feasible topology is replaced by the single required offspring from which it was abstracted, and the optional components in the feasible topology are replaced by all offspring components. Thus, the required component S is replaced by the required offspring S_1, the required component M is replaced by the required offspring M_1, and the optional component T is replaced by both of its offspring, components T_1 and T_2.

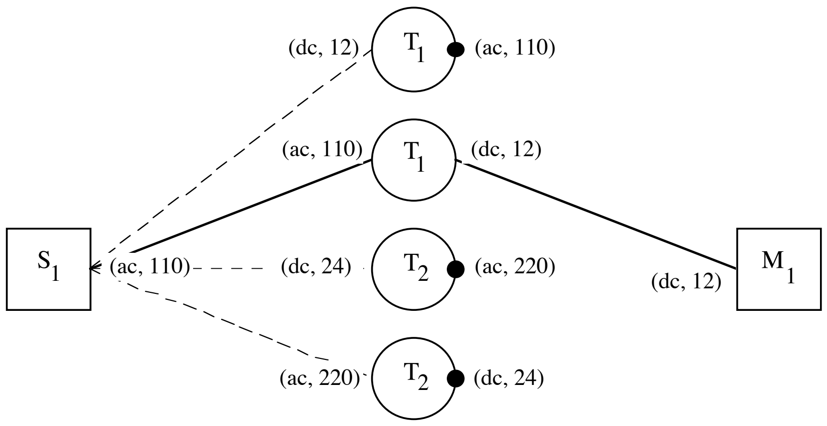

Figure 1.5 Topology Instantiation

In the ABCON process illustrated above, the topology generation process expanded the search tree along four different branches, and the topology instantiation process expanded the search tree along five different branches. Thus, the total ABCON process required nine expansions, while the explicit enumeration methodology required twelve expansions. While this 25 percent reduction in computation is fairly modest for this simple example, the remainder of this thesis is devoted to showing that the reduction can be substantial.

The ABCON methodology is implemented in Common LISP, with source code listed in [Snavely and Papalambros 1992]. The ABCON methodology has been applied to strictly pre-parametric design in domains where the individual elements or building blocks can be accurately represented by independent, idealized components. Just as in the example above, numerical port specifications are simply symbols, used like any other symbol for purposes of pattern matching. While this restriction might seem imposing, one of the goals of this thesis is to prove that a significant amount of meaningful conceptual configuration design can be performed without involving numerical propagation.

1.4 Outline of the Thesis

The thesis is organized as follows. Chapter 2 reviews much of the published literature related to computational configuration design. This literature review is divided into two sections; the first section reviews the research that is directly relevant to the ABCON methodology, and the second section reviews the research that has indirectly impacted ABCON development. Chapter 3 formally defines the terminology and illustrates the concepts that are fundamental to the ABCON methodology. Chapter 4 formally presents the ABCON methodology and illustrates each aspect using a common design example. Both Chapters 3 and 4 are written in a precise, formal manner in order to avoid any ambiguity. Chapter 5 explores the effectiveness of the ABCON methodology by empirically comparing ABCON to an explicit enumeration methodology. Although the ABCON methodology is significantly better than the explicit methodology, ABCON does not overcome the combinatorial explosion inherent to configuration problems, leading to the conclusion that heuristics are required. Chapter 6 presents a heuristically-modified ABCON methodology, called H-ABCON, and Chapter 7 applies the H-ABCON methodology to a number of design problems in several diverse domains. Finally, Chapter 8 details the conclusions drawn from this work and suggests ways in which the ABCON and H-ABCON methodologies could be further expanded.