A Bond graph based approach to Case-based synthesis

T.N.Madhusudan

July 1995
CMU-RI-TR-95-29

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The paper presents an algorithm for case-based synthesis of single-input single-output power-drive systems based on dynamic behavior design specifications. The case-base consists of electro-mechanical components modelled using the bond-graph formalism. An hierarchical classification scheme for devices that distinguishes between the functional and structural aspects of a device is described. The synthesis algorithm also combines conceptual and parametric design in a coherent computational scheme. Examples illustrating the device representation and synthesis algorithm are described.

This research is sponsored by the Office of Naval Research under contract no. ONR N00014-92-J-1298.

Keywords: Case-based design, Bond graphs, conceptual design

Contents

List of Figures

List of Tables

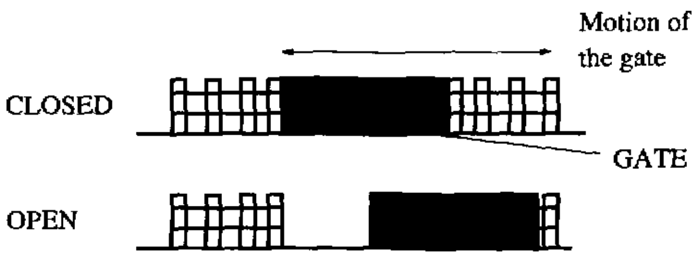

Figure 1: A driveway gate

1. Introduction

1 The engineering design process has been classified into four stages, namely, (1)*Conceptual design* wherein physical concepts and engineering principles are used to generate prototypes that are *expected* to meet given design specifications, (2) *Parametric design* wherein geometry, form and material parameter values are chosen for each feasible conceptual prototype, (3) *Configuration design* involving spatial arrangements and sizing of the components of the synthesized prototype and finally (4) *Embodiment design* wherein detailed specifications for the product are generated aided by thorough engineering analysis. Successful computer-design aids have been built to support parametric and configuration design such as structural optimization and layout tools. Research in developing models for design computation has primarily focused on providing tools for analysis of designed artifacts and their computerized representation as solid models. Only limited work has been performed to provide support for design synthesis at the early stages of design. Tools to support conceptual design have been limited in their expertise because of the variety of physical concepts that need to be represented. It is also unclear how the conceptual design phase interacts with the parametric and configuration design stages. An example design problem that illustrates the different design stages and highlights the need for tools that aid conceptual design is described in the following section.

1.1. An example problem

Consider the synthesis of an electro-mechanical drive system that opens and closes a gate to the driveway of a house as shown in Figure 1. Given a signal to open the gate, the drive system moves the gate horizontally on guide-ways in a particular direction; the drive system moves the gate in the opposite direction when given the signal to close the gate. Design specifications for the drive system are the frictional force that resists the motion of the gate, the speed variation of the gate, the input power supply to the drive system i.e. the input electrical voltage and current availability and variation with time. A drive system for the gate can be assembled from numerous off-the-shelf components such as motors, gears, linear slides and other mechanisms to meet these specifications,. Four possible designs of increasing complexity are shown in Figure 2. In each design, the thick arrows denote the direction of power flow. Energy from the electrical source flows through each component

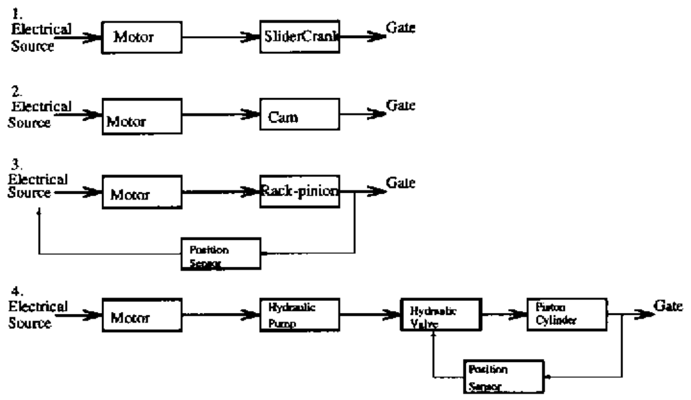

Figure 2: Possible designs for gate drive system

and is finally converted into work moving the gate. The schematic diagram for each design denotes the *power connectivity* structure between different components (notated as boxes) and is called the *device topology* of the design. The choice of AC or DC motors in the device topologies is dictated by the nature of the electrical source. Designs 1 and 2 are simple designs wherein the gate opens, stays in the open position for a brief period of time and then closes. Designs 3 and 4 involve feedback about the position of the gate denoted by the thin arrow. A position sensor detects the limiting positions of the gate and triggers a directional switch to change current direction in design 3 and the bi-directional valve position in design 4. Each component in the device topologies in Figure 2 has a well-defined dynamic behavior and role to play in the over-all functioning of the design. For example, the electrical motor converts electrical power into rotary mechanical power and a cam converts rotary motion into reciprocatory motion. The choice of a design from Figure 2 also depends on metrics such as the cost, the weight, the spatial volume and reliability of each component in the design and the overall assembly. A device topology is generated as a result of a sequence of design decisions. We illustrate the design choices for generating the gate-drive system device topologies in Figure 3. The process of choosing the correct type of off-the-shelf components i.e. a four-bar mechanism vs. a rack and pinion mechanism and combining these components in a feasible manner that provides the required functionality is *conceptual design*[1, 2]. The different motor and mechanism combinations shown in the figure provide *conversion of rotary motion to translatory motion*. Once a particular combination of components is chosen, the next issue is sizing these components i.e. choosing a large or small motor with large or small rack-pinion mechanism. This process is called *parametric design*. This is illustrated in the figure by the different motor and rack-pinion combinations. An associated step is the process of choosing spatial orientations for each component and forming the overall shape and size of the drive system for the gate. This is called *configuration design*. As shown in the figure, the motor and rack-pinion can have a number of *relative orientations* depending on spatial constraints in the design specification. The foregoing example describes the synthesis of new devices in contrast to the task of *routine design* wherein one is primarily involved in parametric design involving resizing of components given a particular device topology. Choices of components and device topologies in the conceptual design stage may not satisfy parametric or configuration requirements and the process of design has to be

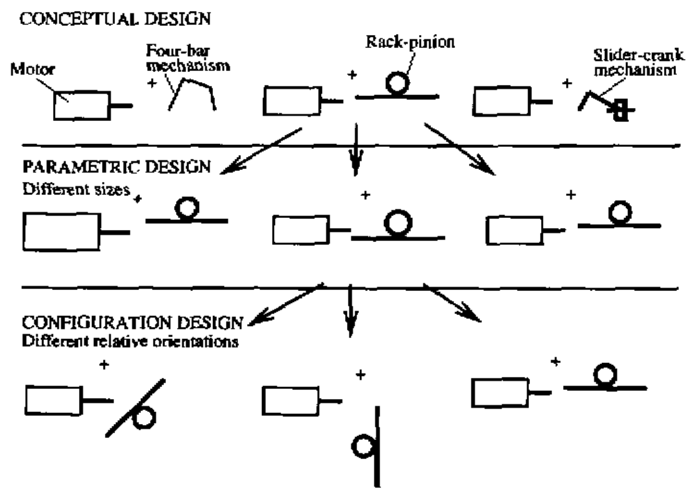

Figure 3: Design process for the gate drive system

repeated with a different initial choice of concepts thus leading to increased product design cycle times and cost. *Computational design aids that provide the human designer the ability to generate different initial concepts i.e. components and valid component topologies given a set of design specifications and also explore the interaction effects of conceptual, parametric and configuration choices will be invaluable.* Such aids would help in the choice of off-the-shelf components and their consequent assembly without recourse to synthesizing new devices from scratch. In this paper, we present a computational scheme to generate valid assemblies of components given design specifications and also perform parametric design on the synthesized assemblies.

Computational aids for supporting design must exhibit certain essential characteristics. Since the space of feasible designs is vast, the synthesis algorithms must generate all designs on request or some valid designs depending on the nature of the design specifications. The design algorithms must also consider measures such as cost, reliability, robustness etc. to trade-off various design alternatives and propose only those solutions that satisfy required design criteria. For more detailed design specifications, the synthesis routines must provide design solutions that contain the same or more amount of design detail. The design algorithms must reflect a *domain-independent* approach such that an algorithm for designing combinations of mechanical devices can be adapted to synthesize hydraulic devices.

Finally, in the case of electro-mechanical devices, the devices are described at multiple levels of detail and abstraction. An example of two abstract representations for an electro-mechanical component are the solid model representation and the dynamic differential equation representation of the device. The representations support different kinds of physical information relevant for design; the solid model provides information on shape, volume, weight etc. while the differential equation provides information on the dynamics of the device. Therefore the synthesis procedure must be able to process design information at and across different levels of abstraction. Design is a generative task wherein a variety of feasible designs are created and evaluated. Design is computationally more expensive than analysis of a given design alternative based on a given physical model of the device. Also the information available during the generative process is rather incomplete at times and the variety of alternatives for each design decision is rather large leading to combinatorial explosion of choices. Therefore the algorithms must make efficient use of partial knowledge to reduce search.

Conceptual design systems such as IBIS [3] uses a graph-based representation for modeling device topology but fails to capture time-dependent behavior of physical parameters. Qualitative representations of device behaviors are incomplete with respect to device structure[4] and also the dynamic description do not enable combination of components. The substance-behavior-function model [5] model does not capture physical processes but is more a descriptive model of devices. Kota [6] has proposed the qualitative motion synthesis approach which focuses on kinematic design of mechanisms. A qualitative model and a matrix-method to combine different qualitative motion descriptions is used to build devices. A rule-based approach is provided in [7] for synthesis of mechanisms. A predicate-logic representation is used to model devices and a complex search procedure to compute new designs. Other conceptual design schemes are described in [2]. In summary, the design methods proposed so far have not utilized both *topological information* about devices and their *constituent physical process relationships.* From the viewpoint of the staged design process, no computational scheme with a well-defined device model has been proposed to *integrate conceptual, parametric and configuration design* stages in a feasible manner.

A computational methodology for design that combines generative aspects of design and also combats the computational inefficiency in a feasible manner is *Case-based design*[8]. Case-based reasoning is a paradigm that aims to use experiential knowledge gained in solving previous problems to formulate solutions for new problems[9]. Primary elements of a CBR system are the case-base, wherein previous problems, their solutions, models etc. are stored. and an inference mechanism that uses the information stored in the case-base. The inference mechanism converts the given problem specifications into relevant indices for retrieving cases from the case-base, retrieves relevant cases, validates the retrieved cases as plausible solutions and if necessary modifies some of the cases to meet the new problem specification and proposes a new solution. Case-based design provides for use of previous designs and fragments of complete designs as partial solutions in the synthesis process. Having access to previous designs reduces the complexity of the search in the ill-structured domain of electro-mechanical design. Cases provide coherent models of devices across multiple abstractions enabling consistent design reasoning across different representations of a device. Since a case captures all relevant design information, it provides for performing conceptual, parametric and configuration design in an integrated manner. Cases also provide a means to *operationalize* design knowledge either as rules or as descriptive data structures. Cases also provide a sort of *feasibility check* on new designs by providing information regarding success and failure on previous designs. with respect to different design alternatives. Another interesting aspect is that previous cases provide designers information with regard to the physical realizability of various designs i.e. guarantee that devices can be manufactured with reasonable investments of capital and time. The CADET [10, 8] system is based on the qualitative reasoning framework and uses Influence graph diagrams(ISD) for modeling devices. A graph-based indexing scheme is used to retrieve cases. IDEAL[11] and Kritik[5] are other CBR-base systems of note for synthesis of mechanical devices based on the SBF device models.

A CBR based design methodology raises the issue of the definition of a design case and how a design case is to be represented. A design case representation must correspond to the various models of the physical world phenomena. We cannot simply structure a case in terms of axiomatic logic-theoretic representations. The models of the physical world in engineering and physics are non-linear and stochastic. The device models also involve quantitative and temporal variations. A rule-based approach is not flexible enough to handle these different aspects of the physical world in a consistent manner and is primarily incomplete. Design reasoning to a certain extent involves interpretation of these complex physical models, studying the behavior of these models when various aspects of these models are tweaked and choosing the right combination of these physical models to create feasible designs that can meet the design specifications. Therefore a case-representation requires a feasible combination of axiomatic and analog models of physical phenomena. Further, since synthesis involves combinations of components, procedures for aggregating and combining cases need to be defined. Combinations of cases must satisfy all physical conservation and thermodynamic laws. A principled way of combining cases and ensuring feasibility of the combined design is required. It is also imperative that the combination of cases be feasible at all levels of abstraction.

We have developed a case-based design methodology that addresses the different issues raised in the foregoing paragraphs. Our methodology combines bond-graph based device models to meet design specifications in a systematic manner. In our CBR-model of design, the case-base consists of device models of components and assemblies of components. The steps in the inference procedure are shown in Figure 4. Design specifications from the user are transformed into indices for retrieval of cases (devices) from the case-base. Cases are retrieved and composed into an assembly. Each synthesized assembly is then validated. Designs can be validated via simulation or through the use of validation rules. Successful designs as well as failures are archived in the case-base. The case-base consists of cases that store design information related to both conceptual and parametric design. The conceptual and parametric design tasks are interleaved in the inference mechanism. An interesting feature to note is that the CBR mechanism provides an explicit inference step for *assembly* of retrieved components i.e. to perform synthesis. The cognitive model of the CBR process has the notion of retrieving different relevant cases from memory and *adapting* those cases to propose a new solution. *Composition* of cases is one of the many available adaptation schemes and plays a critical role in synthesis of assemblies from components.

Bond-graph based representations have been used for design in [12, 13, 14, 15, 16]. In [12],the synthesis procedure generates a network of bondgraph elements with bond-graph primitives such as TF,GY,R,I, C elements and 0-1 junctions to span the input and output bond-graph chunks, based on rules that aim to generate differential equations of a prespecified order. An exponential search procedure is used to generate bond-graphs for electromechanical single-input single-output systems. In [13], bond graphs are viewed as a defining

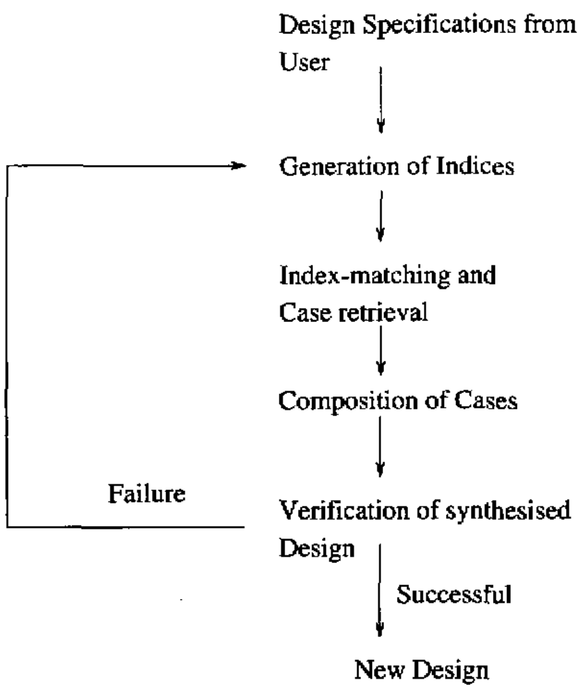
Design Specifications from User
Generation of Indices
Index-matching and Case retrieval
Composition of Cases
Failure
Verification of synthesised Design
Successful
New Design

Figure 4: The case-based design process

a a grammar and synthesis rules that transform the bond graphs are defined. We extend the usability of bond-graphs as a device representation by addressing device behavior over time in the design specifications. We follow a systems-theoretic approach and do not aim to synthesize embodiments of the devices but identify the components and their connectivities [17, 1]. Bond graph based design schemes are reviewed in detail in [18]. A typology of devices has been developed based on device input-output relations that capture dynamic behavior. Further, these input-output relations are related to the structural aspects of the device such as geometry and other material properties. We have enhanced the role of bond graphs by observing that lumped parameter coefficients in bond-graph relations provide a convenient link to study the effects of device structure on the device behavior. For example, the bond graph model of a pair of spur gears is a mechanical power transformer with the tooth ratio of the gears in mesh as a modulating parameter. The ratio of the teeth is the only structural parameter that affects the ideal dynamic behavior of the device with respect to its intended role of transforming power. This tooth-ratio provides the link to the structural aspects of the gear. In the foregoing example, the gear-ratio required can be obtained by a variety of teeth ratios. Other attributes of the gear such as size, teeth geometries and material properties do not affect "ideal gear behavior". This modeling abstraction facilitates reasoning across device dynamic behavior and device structure. We hasten to note that though bond-graph analysis recognizes the existence of these lumped parameters, they have been considered as *given* i.e. fixed for purposes of analysis and not as variable entities i.e. entities which are a design choice. The case-representation consists of a bond-graph based device model that captures relevant physical effects and an axiomatic model that aids interpretation of the device model and captures parametric and geometric attributes of the device. This case representation allows consistent reasoning across device behavior and structure. Also the representation provides for monotonic behavior of the synthesis procedure i.e. more detailed specifications generate detailed but a smaller number of solutions. The device representation allows for combining components or cases. Each component is modelled as a physical system with a set of input and output *ports*. A port is a conduit for transfer of energy (power) into and out of the component. When components are assembled, they are connected at their ports. The output port of one component becomes the input port of the adjacent component. When two devices are connected at their ports, the power and energy variables, such as force, velocity and momentum, at each port are constrained to be equal. Power flows from one device to another at any instant of time. Power flows via the first component into the adjacent one as shown in Figure 2. An assembly of components can be modelled as a single physical system with its own input and output ports. The input-output relations of the components can be combined to determine the overall input-output relation for an assembly. Thus given a input-output design specification, one can verify whether a proposed assembly of components can meet the given design specifications. The verification procedure thus provides a *stopping criteria* for the generative search process by ensuring that non-feasible design alternatives are not further explored. The bond-graph model captures the ***physical effects*** in a device and cannot be put into a *one-to-one* correspondence with physical components of a device. The port models of devices are idealized mathematical versions of real physical embodiments such as masses, springs, and gears. Complex two-port systems can be modelled by assembling these two-port models serially. Multiple input-output systems can be assembled from one, two, three and other multi-port components. The design methodology enables the synthesis of assemblies of multiple input-output power and signal devices.

In this paper, we focus on the synthesis of assemblies consisting of components with one-input and one output port to illustrate the synthesis procedure. We describe a computational scheme that combines both the conceptual design and parametric design tasks using the above representation. Configuration design tasks are not addressed. In section 2, we present the models of devices and their representation. We also describe the possible types of design specifications that are entailed by this device model. In Section 3, we present the synthesis algorithm, in Section 4, we present domain heuristics with respect to the algorithm and the device representation and conclude in Section 5.

2. Device representation

Our proposed device representation captures physical phenomena as energy interactions and device structure (topology) as a directed graph. Device functionality is dependent upon the ability of the device to transform either power or signal flows through the device. Each such power or signal transformation is provided by some physical effect that is encapsulated by the device. The dynamic behavior of an assembly of components is dependent on the device topology and the transformation behavior of each component. In the following sections, we describe the possible power transforming behaviors of components and present a schema representation of power devices.

2.1. Energy interaction models for devices

The mathematical model of energy *interactions encapsulated* by devices is based on the bond graph formalism [19, 20, 21]. The bond graph formalism identifies three types of energy interactions among devices. The energy behaviors of devices are *energy storage, energy dissipation and energy transmission.* Complex device behavior arises when components with storage, dissipation and transmission behavior are assembled together. The dynamics of physical devices are derived by the application of *instant-by-instant energy conservation.* In the bond-graph formalism, devices are modelled by components connected at places where power can flow between the components. Such places are called *ports* and devices with one or more ports are called *multiports. Energy storage and dissipation behavior* is exhibited by devices with *one* power port. Devices such as springs, resistors, masses etc. can be modelled as one-port devices. *Energy transmission* behavior is exhibited by devices with multiple input and output ports. A tee-pipe can be modelled as a multi-port device. In this paper, we focus on synthesis of single input, single output devices called two-port devices. Devices such as motors, slider-crank mechanisms, cams etc. in Figure 2 are two-port devices. The overall gate-drive system in the earlier example can be envisioned as a two-port device wherein electrical power flows in at the input port and is used to mechanically translate the gate at the output port. We summarize energy transmission behavior of a two-port device in the following paragraph.

Each power port of a device has four variables, namely, *effort* ($e(t)$), *flow* ($f(t)$), *effort integral* ($\int e(t)dt$) denoted as $\mathcal{E}(t)$ and *flow integral* ($\int f(t)dt$) denoted as $\mathcal{F}(t)$. The *power* ($P(t)$) is equal to $e(t).f(t)$. $e(t)$ and $f(t)$ are called *power* variables. The energy flowing through a port over a period of time $E(t)$ is given by $\int e(t).f(t)dt$ or $\int f(t).d\mathcal{E}(t)$ or $\int e(t).d\mathcal{F}(t)$. $\mathcal{E}(t)$ and $\mathcal{F}(t)$ are called *energy variables.* A power port in a device belongs to an energy domain. Power and energy variables can be identified for electro-magnetic (EM), mechanical translation (MT), mechanical rotation (MR), thermal (TH) and hydraulic (HY) energy domains and are listed in Table 2.1. The first column lists the energy domain and the ensuing columns the effort, flow, power, effort integral, flow integral and energy variables of that energy domain. Energy domains that involve radiative transfer of energy (solar, light, acoustics and radiated heat energy) are not modelled though successful attempts have been made to extend the bond graph methodology to radiative phenomena.

For a two-port device, at every instant of time $e_1(t).f_1(t) = e_2(t).f_2(t)$, where the subscript 1 denotes input port and 2 denotes output port. The above equation implies that in a two-port system whatever power is flowing into one side of the 2-port is simultaneously flowing out of the other side. To satisfy the power conservation relation in a physical two-port system, e_1 may be related to e_2 and f_1 may be related to f_2. Another possibility is e_1 may be related to f_2 and f_1 may be related to e_2. Each set of two relations is called a *device relation.* Table 2 shows commonly occuring device relations in a variety of two-port power components. Each relation in Table 2 is denoted D_i where i indicates the serial number of the relation in the table. The term k denotes an algebraic coefficient in the device relations. $\mathcal{F}$ and $\mathcal{E}$ denote time integrals of effort and flow parameters. There are numerous other

Domain	Effort	Flow	Power	Effort Integral	Flow Integral	Energy
Mech. trans.	Force (F)	Velocity (v)	F.v	Momentum	Displacement (x)	Mech. Work (W)
Mech. Rotation	Torque (T)	Angular Velocity (ω)	T.ω	Angular Momentum	Angle (Θ)	Mech. Work
Hydraulic	Pressure (P)	Flowrate ($\dot{Q}$)	$P.\dot{Q}$	Pressure Momentum	Volume (V)	Hydraulic Energy
Electro-Magnetic	Voltage (e)	Current (i)	e.i	Flux (λ)	Charge (q)	Electrical energy
Thermal	Temperature	Entropy-flow	*	*	*	*

Symbols in parenthesis denote conventional symbolic names for physical parameters.

Table 1: Energy and Power parameters

No.	Relations
1	$e_2 = k.e_1, f_2 = f_1/k$
2	$e_2 = e_1/k, f_2 = f_1.k$
3	$e_2 = k.f_1, f_2 = e_1/k$
4	$e_2 = k.e_1.Sin(\mathcal{F}_1), f_2 = f_1/k.Sin(\mathcal{F}_1)$
5	$e_2 = e_1/(k.Sin(\mathcal{F}_1)), f_2 = f_1.k.Sin(\mathcal{F}_1)$
6	$e_2 = k.e_1.Sin(\mathcal{F}_2), f_2 = f_1/k.Sin(\mathcal{F}_2)$
7	$e_2 = e_1/(k.Sin(\mathcal{F}_2)), f_2 = f_1.k.Sin(\mathcal{F}_2)$
8	$e_2 = k.f_1.Sin(\mathcal{F}_1), f_2 = e_1/k.Sin(\mathcal{F}_1)$
9	$e_2 = k.f_1.Sin(\mathcal{F}_2), f_2 = e_1/k.Sin(\mathcal{F}_2)$
10	$e_2 = f_1/(k.Sin(\mathcal{F}_1)), f_2 = e_1.k.Sin(\mathcal{F}_1)$
11	$e_2 = f_1/(k.Sin(\mathcal{F}_2)), f_2 = e_1.k.Sin(\mathcal{F}_2)$
12	$e_2 = k.e_1.Sin(\mathcal{E}_1), f_2 = f_1/k.Sin(\mathcal{E}_1)$
13	$e_2 = k.e_1.Sin(\mathcal{E}_2), f_2 = f_1/k.Sin(\mathcal{E}_2)$
14	$e_2 = e_1/(k.Sin(\mathcal{E}_1)), f_2 = f_1.k.Sin(\mathcal{E}_1)$
15	$e_2 = e_1/(k.Sin(\mathcal{E}_2)), f_2 = f_1.k.Sin(\mathcal{E}_2)$
16	$e_2 = k.f_1.Sin(\mathcal{E}_1), f_2 = e_1/k.Sin(\mathcal{E}_1)$
17	$e_2 = k.f_1.Sin(\mathcal{E}_2), f_2 = e_1/k.Sin(\mathcal{E}_1)$
18	$e_2 = f_1/(k.Sin(\mathcal{E}_1)), f_2 = e_1.k.Sin(\mathcal{E}_1)$
19	$e_2 = f_1/(k.Sin(\mathcal{E}_2)), f_2 = e_1.k.Sin(\mathcal{E}_1)$
20	$e_2 = k.e_1.\mathcal{G}(\mathcal{F}_1), f_2 = f_1/k.\mathcal{G}(\mathcal{F}_1)$
21	$e_2 = \mathcal{H}(e_1), f_2 = f_1.e_1/\mathcal{H}(e_1)$

Table 2: Two port device relations

No.	Algebraic coefficients	Lumped parameters	Relation
	in device relations	of device	
1	k_1	p_1	$k_1 = p_1$
2	k_1	p_1	$k_1 = p_1^2$
3	k_1	p_1, p_2	$k_1 = p_1/p_2$
4	k_1, k_2	p_1, p_2	$k_1 = p_1, k_2 = p_2$
5	k_1, k_2	p_1, p_2	$k_1 = p_1 + p_2, k_1 = p_1/p_2$

Table 3: Common structural relations

types of device relations possible as long as they satisfy the law of conservation of energy such as relations D_{20} and D_{21}, where $\mathcal{G}$ and $\mathcal{H}$ denote functions such as exponential, log etc. Each two-port power device encapsulates a device-relation and the generalized effort and flow variables at each port are instantiated based on the energy domain of the ports of the device. A pair of gears is modelled by $(e_2 = k.e_1, f_2 = f_1/k)$, where the input and output energy domain is mechanical rotation. An induction motor is modelled by $e_2 = k.f_1.Sin(\mathcal{F}_2), f_2 = e_1/k.Sin(\mathcal{F}_2)$ and a DC motor by $e_2 = k.f_1, f_2 = e_1/k$ wherein the input energy domain is electrical and output energy domain is rotary mechanical.

The device relations between the port variables (efforts and flows) have algebraic coefficients, such as k, as shown in Table 2. These algebraic coefficients are functions of lumped physical parameters that depend on the geometry and material characteristics of the device. *Lumped* physical parameters do not have a spatial extent. For example, the concept of a *mass* of a component concentrated at a point in physical space is a lumped parameter approximation though the mass obviously depends on the spatial volume occupied by the component. Relations such as those in Table 2 may have multiple algebraic coefficients which are denoted as k_i. The lumped geometric and material parameters are denoted by p_i. The set of relations that map each k_i of a device as mathematical functions of p_i is called a *structural relation*. The lumped parameters, p_i are called *structural parameters*. A variety of structural relations exist depending on the number of coefficients in a given device relation and the structural parameters of the device. Table 3 shows common structural relations. Each structural relation is denoted as S_i where i indicates the serial number of the relation in table 3. For a pair of gears, the algebraic coefficient k (denoted as k_1), is defined by $k_1 = p_1/p_2$, wherein each lumped parameter $(p_i, i = 1, 2)$, is the number of teeth in each gear. For a DC-motor, the structural relation is $k_1 = p_1$ and the "lumped" parameter is the field current that excites the magnetic coils and the algebraic term in the device relation is the motor constant. Therefore by not simplifying the structural relations, more information about the devices' actual structure can be imported into the bond graph.

Devices exhibit dynamic output behavior when they are driven by power inputs. Devices can exhibit dynamic output behavior in two ways, (1) when the input power flow at the input port varies and (2) when the lumped parameters vary with time. Either of these two changes will manifest itself as variations in the behavior of parameters at the output port. When both kinds of input effects can take place simultaneously, the output dynamic behavior is a superposition of the two input effects. In this paper, we synthesize two-port devices wherein the structural parameters are *not* allowed to vary with time. Every component has *constant* values for its structural parameters. This assumption precludes signal-dependent modulation of dynamic behavior of devices.

2.2. Representation of dynamic behavior of efforts and flows

Design specifications for a required device are primarily specified in terms of corresponding effort or flow time histories at the input and output ports since one does not know the device relation before synthesis. Thus a representation is required for capturing the time-histories of effort and flow parameters at the ports of a device. In this section, we describe the representation of time histories for parameters and also describe its significance in the modeling of devices and also the synthesis process.

Dynamic output behavior is obtained by dynamic variations at the input port of a device. For example, an increase in input power can be obtained by increasing the effort value, flow value or both, thus causing a variation in the output port variables. Each parameter at the input and output port describes a trajectory over time that describes the overall dynamic behavior of the device. Theoretically, an infinite number of such continuous trajectories are possible for a given effort or flow parameter. We model a single time trajectory of a parameter as piece-wise continuous functions of time. Consider the trajectory of displacement of the follower (with the cam axis as reference) during rise of a parabolic cam as shown in Figure 5. The follower has a constant acceleration, linearly increasing velocity and a parabolic displacement over the follower rise time-period t_{rise} and the same behavior in the opposite direction during the follower fall time-period, t_{fall} as shown in figure.

The time trajectory of a parameter is discretised into a sequence of regions, $(T_1, T_2, T_3 ...)$, T_i denotes the ith time interval and T_1 is the interval containing the time origin. For the displacement, velocity and acceleration trajectories for the cam in the foregoing example, there are two distinct regions t_{rise} and t_{fall} which constitute T_1 and T_2. In each region T_i, the parameter trajectory with time is approximated as a closed-form function of time, t. The possible parameter functions of time that we have implemented are shown in Table 4. y can be efforts or flows at the ports of devices and t denotes time. m_i and c_i denote constants. Each parameter-time relation is called a *P-relation* and is denoted P_i where i denotes the serial number in the table. The time trajectory of a parameter over a given period of time can be approximated by a set of *P-relations*. We restrict our representation only to the above relations, since any complex time-history of a parameter can be approximated by the above relations in a piece-wise manner. A parameter trajectory can lie only in the first and fourth quadrants of the parameter-time coordinate system since time is always positive. All physical parameters are scalars. A positive or negative magnitude for a physical parameter denotes the *direction of action* along a given direction in coordinate system chosen by convention. A positive magnitude indicates that the physical parameter acts in the same direction as the given vector and a negative magnitude denotes that it acts in the opposite direction to the

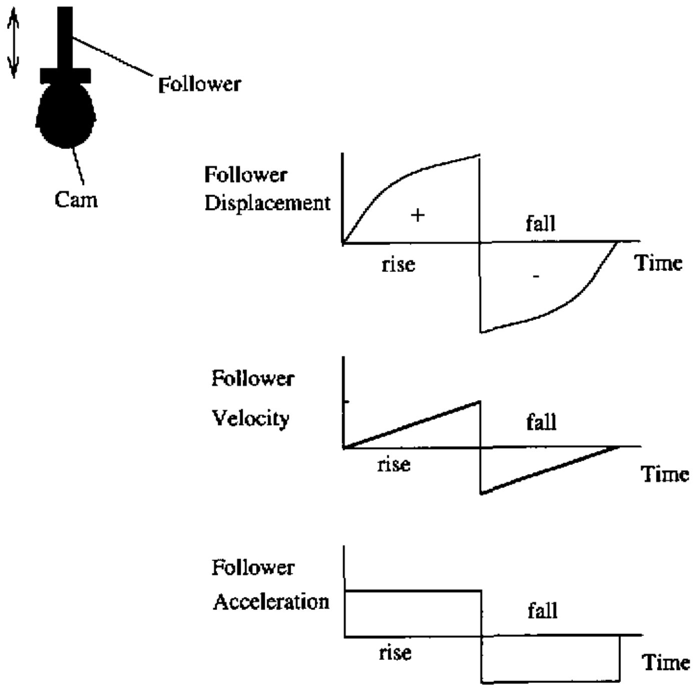

Figure 5: Displacement trajectory for parabolic cam follower

No.	Relation
1	$y = m_1.t + c_1$
2	$y = m_1.Sin(t) + c_2$
3	$y = m_1.Log(t) + c_4$
4	$y = m_1.Exp(t) + c_5$
5	$y = m_1.t + m_2.t^2 + c_6$

Table 4: Parameter functions of time

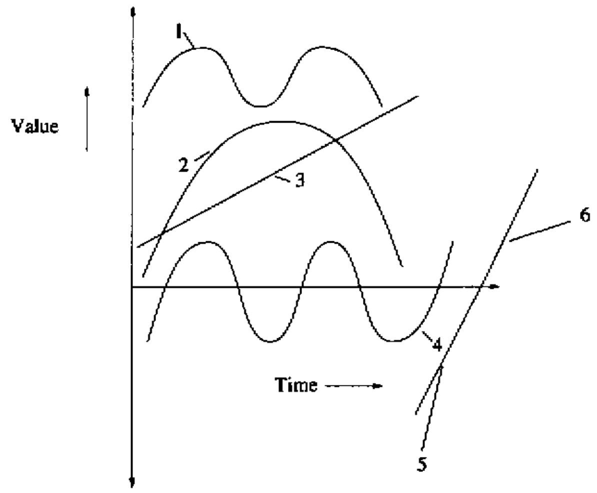

Figure 6: Possible parameter trajectories

given vector. Thus each relation listed in the Table 4 can be used to describe time trajectories as shown in Figure 6. $y = m_1.t + c_1$ can represent trajectory 3 while $y = m_1.t + m_2.t^2 + c_6$ can represent trajectory 2. Each such relation does not have any information regarding the *direction of action* of a physical parameter. For example, $y = m_1.Sin(t) + c_2$ can be used to represent trajectories 1 and 4 in Figure 6 since both are sinusoidal with respect to time. Physically the two trajectories denote two different physical events, trajectory 1 describes a sinusoidal variation in magnitude such as an oscillating pressure pulse while trajectory 4 describes a change in magnitude and direction such as the reciprocating motion of the slider in a slider-crank mechanism. Reciprocation is indicated by the sinusoidal curve in trajectory 4 switching between the first and fourth quadrants. Thus it is essential that we distinguish between the two trajectories, 1 and 4. Knowledge of m and c_i in the parameter relations is not enough as shown by trajectories 5 and 6 in Figure 6. Though the two trajectories lie on the same line and are represented similarly, they denote different physical events. The only information that is needed to discriminate is the direction. Each parameter trajectory is represented by a P_i which is also annotated with directional information. The literal `+` denotes that the parameter relation describes a trajectory that lies only in the first quadrant, literal `-` denotes that the parameter relation describes a trajectory that lies only in the fourth quadrant and literal `+-` denotes that the parameter relation describes a trajectory both in the first and fourth quadrant. Hence trajectory 1 will be annotated with a `+` while trajectory 4 will be annotated by a `+-`. Thus a parameter trajectory in a discretised time region is denoted by a list *(parameter relation, quadrant, time duration)* wherein *parameter relation* is a triplet $(snum, c, m)$ where *snum* is the serial number from Table 4, m and c are lists of coefficient values for the relation; a *quadrant* is one of the literals `+`,`-` or `+-` and time duration is a pair (s_t, f_t) where s_t and f_t denote start time and end times for trajectories. The follower acceleration trajectory in Figure 5 over the rise and fall periods will be represented as shown in Table 5. `trise` and `tfall` denote rise and fall times. The first column lists the time-region and the second column the parameter relation, quadrant and time durations. In each region, the linear constant velocity is represented by $y = m_1.t + c_1$ (denoted by the first

Time-region	Trajectory representation
T_1	`(1,(c1),(m1)), +, (0 trise)`
T_2	`(1,(c2),(m2)), -, (trise trise+tfall)`

Table 5: Velocity trajectory for parabolic cam follower

element 1). Specific trajectories are defined when the constants m_i and c_i are defined. Thus in Table 5, $m_1 = 0, m_2 = 0, c_1 = 1$ and $c_2 = 1$ describes a constant velocity of 1 m/s over the follower rise and fall periods. The different quadrants for the velocity trajectory during rise and fall are denoted by the literals + and -. The trajectory of follower displacement during t_{rise} and t_{fall} can be represented by $y = m_1.t + m_2.t^2 + c_6$ i.e. parabolic with appropriate coefficients.

Thus to describe the dynamic behavior of a device over a given time-period, one would discretise the time-period into discrete regions and in each region, describe the behavior of input and output port parameters as functions of time. Thus for a two-port device with one discretised time region, four parameter trajectories, one each for the input and output port efforts and flows are required to completely describe the device behavior. Usually a physical device that encapsulates a device relation does not produce outputs for all kinds of input effort and flow. Only certain kinds of parameter relations are allowed for the input effort and flow. For example a DC-motor will not produce a rotary torque if given a sinusoidal input voltage. A device relation for modeling a device is also annotated with all the possible inputs it can handle. From the fifteen possible trajectories for input port effort and flow i.e five types of P_i and three directional quadrants, the valid trajectories are listed in the device representation.

Why is such a complex representation of time-history behavior required ? First, the design specifications for a device are given in terms of nominal corresponding time-histories for the efforts and flows at the ports of device. For example, design specification for a motion generating mechanism is in terms of requisite time-profiles for the output linkages. Secondly, the constitutive device relations described in the previous section can produce a variety of output effort and flow time histories depending on the input effort and flow time histories. In theory, therefore a single device can be driven by an infinite number of different types of inputs. Representing a device by cataloging all its input-output parameter time-histories is cumbersome. Therefore though the device relation is a succinct representation of the device input-output relation, a representation for describing time-histories is required to enable choices of devices that may provide by requisite behavior. This choice can only be made by driving the device with the given input time-history and observing the output behavior. We note therefore, that for a given time-history specification, an infinite number of devices or their combinations can be proposed as feasible solutions. Synthesis therefore involves generating combinations of devices and testing them for different inputs as specified in the design specifications. If requisite output behavior is obtained, the new design is accepted. From an algorithmic viewpoint, one needs to match the different input-output

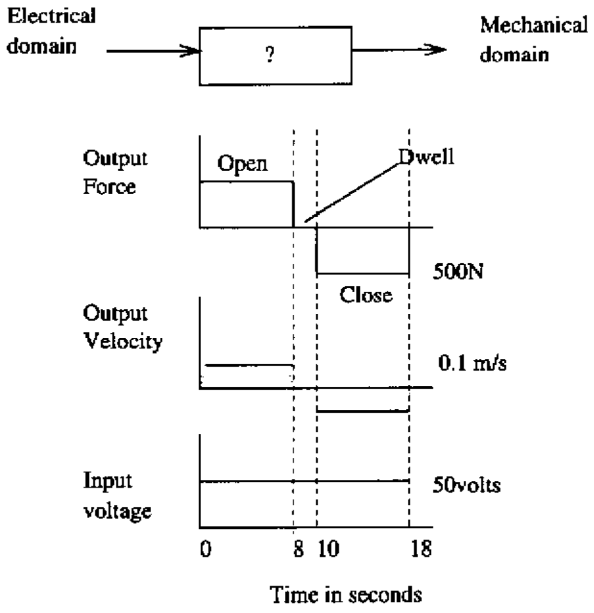

Figure 7: Design specifications for gate drive system

time-history specification to all possible device relations that can provide the requisite input-output transformation. In the following section we describe the use of the above-described parameter trajectory representation for providing design specifications regarding dynamic behavior of a two-port device.

2.3. Design specifications

Design specifications outline the *input and output port energy domains and trajectories for effort and flow at either port over a given time duration.* We illustrate representation of design specifications with the gate-drive system example. In Figure 7 the gate-drive system is shown as a black-box whose input and output energy domains are electrical and mechanical translatory domain. The trajectories for the input voltage, output force required to move the gate and the velocity of the gate are also shown. The output force is a constant (500N) and changes direction from each half-cycle of opening and closing the gate. The velocity of the gate starts from zero and reaches a steady value (0.1 m/s) and then reduces to zero over each half-cycle. There is a well-defined dwell period (2 seconds) between each half cycle (8 seconds) when the gate remains open or closed as the case may be. Actually this dwell period could be large (open gates) but for illustrative purposes we have chosen 2 seconds. The velocity behavior can be approximated as a constant velocity neglecting the quick rise to steady value. The parametric values for force and the velocity are known and specified as shown in the figure. The input voltage is a DC-voltage which is held at a constant value of 50 volts. The input current profile is not fixed as it depends on the load that is driven and is allowed to be any trajectory that meets the load requirements. The trajectories have been discretised into three distinct regions as shown in the figure. The first time region (gate

Attribute	Value		
Input port energy domain	EM		
Output port energy domain	MT		
	Time-region I	Time region II	Time-region III
Input effort trajectory	`((1 50 0) + (0 8))`	`((1 50 0) + (8 10))`	`((1 50 0) + (10 18`
Input flow trajectory	`((* * (0 8))`	`(* * (8 10))`	`(* * (10 18)))`
Output effort trajectory	`((1 500 0) + (0 8))`	`(- - (8 10))`	`((1 500 0) - (10 1`
Output flow trajectory	`((1 0.1 0) + (0 8))`	`(- - (8 10))`	`((1 0.1 0) - (10 1`

* denotes value is unspecified, - denotes that value is null.

Table 6: Design specification representation for gate drive system

opening) lasts from 0 to 8 seconds, the dwell period lasts from 8 to 10 seconds and the third time region lasts from 10 to 18 seconds. The design specification representation for the gate-drive system is shown in Table 6. Each parameter trajectory in a region is approximated by the the relation $y = m_1.t + c_1$ with $m_1 = 0$ (a constant linear trajectory) and c_1 equal to their parametric value. Thus $c_1 = 50$ volts for the input effort, $c_1 = 500$ Newtons for the output effort,$c_1 = 0.1$ m/s for the output flow and c_1 is unconstrained for the input flow. Since there are three discretised regions over a complete cycle of the gate operation, there are three trajectory specifications for effort and flow parameters at each port. Based on these design specifications, the synthesis procedure must identify a case or a combination of cases that converts the input port parameter variations into parameter variations at the output port. If multiple cases are composed, the procedure must also identify the topology in which the cases are connected i.e. the connectivity between the different ports of the components based on the design specifications.

Thus far our discussion has addressed the representation of components. A component device is defined by a single device relation. An assembly of components is defined by each component device relation and the topology of the assembly. Based on the topology of the assembly and the component device relations, it is possible to obtained a closed-form device relation for an assembly. Consider design 1 in Figure 2. The motor is represented by device relation $e_2 = k.f_1, f_2 = e_1/k$ and the slider-crank mechanism by $e_2 = e_1/(k.Sin(\mathcal{F}_1)), f_2 = f_1.k.Sin(\mathcal{F}_1)$. The overall device relationship obtained by eliminating the common port variables is $e_3 = f_1.k_1/k_2.Sin(\mathcal{F}_2)$ and $f_3 = e_1.k_2.Sin(\mathcal{F}_2/k_1)$ where the port parameters are as shown in Figure 8. This is equivalent to device relation D_{11}. An assembly of component devices may or may not have a unique closed-form device relation as listed in Table 2. The device relation for such assemblies can be inferred from the device relations of the components. The structural relation for an assembly of components is the combined set of all structural relations of its components. The dynamic behavior of an assembly can be changed by changing the structural parameters of any of its components. Also, it is fairly obvious, that the input and output effort and flows for an assembly can be described by the parameter trajectory representation developed in the earlier sections. The connectivity

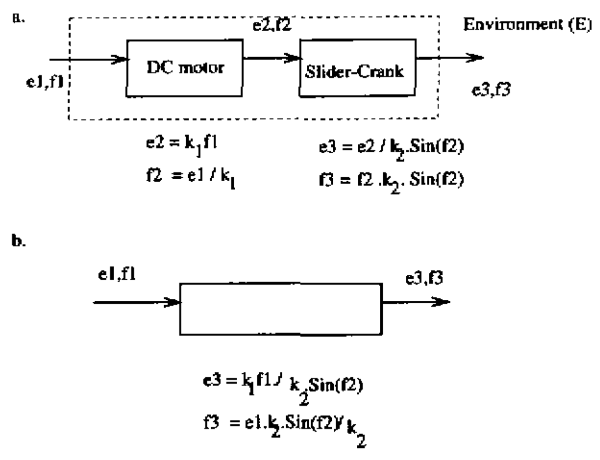

Figure 8: Device relationship for an assembly of components

between components in an assembly is represented by a directed acyclic graph where the directed arcs denote power flow path and the nodes denote components. Thus the device topology of assembly 1 in Figure 2 will be `E` $\rightarrow$ `Motor` $\rightarrow$ `Slidercrank` $\rightarrow$ `E` where `E` denotes the environment which is external to the assembly. Complex devices can be built by combining both components and sub-assemblies. This representation provides a uniform representation for both components and sub-assemblies.

This concludes our discussion on modeling device behavior and its representation. Essential features of the device representation are as follows:

- Devices (components and their assemblies) are modelled as entities that allow transmission of power through conduits called ports.
- Device input-output behavior is modelled by device relations.
- Device dynamic behavior is linked to the structure of the device through structural relations.
- The time histories of the efforts and flows at the ports is modelled by parameter relations.

The representation thus provides a convenient way of describing device behavior in terms of its input-output relation or the nature of the trajectories of its port parameters. Further since the device physics is modelled based on the bond graph formalism, it ensures that combinations of components will be physically feasible without violating any physical laws. Further the port-models of devices provide a convenient means to composing assemblies by connecting components at their ports. In the following subsection, we use the above-described device model to represent components and assemblies as *cases* in a case-base.

2.4. Cases and case-base organization

The content and organization of the case-base determines the validity of the solutions and efficiency of the retrieval algorithm in CBR-based design systems. In our implementation, the case-base stores a variety of devices encapsulating different device relations and structural relations. A device relation defines a class of devices that exhibit similar dynamic behavior i.e. the input-output relationship. A particular physical realization (instance) of a device, belonging to the class defined by a given device relation, is defined when the structural relation is defined and values for the structural parameters are chosen. A spur-gear pair with gear-ratio two is defined when a gear with forty teeth is meshed with a gear with twenty teeth. It is of interest to note that a device relation can be obtained by a variety of physical realizations each with different structural relations or with different values for the lumped structural parameters. For example, a gear-ratio of two can a be obtained either by choosing helical gears that provide a gear-ratio of two or by meshing a spur gear with twenty teeth with a ten teeth spur gear. Each case in the case-base is a well-defined instance of a device relation and structural relation. Thus if a case exists in the case-base it is physically realizable.

Each case is represented as a schema with the attributes and possible values as shown in Table 7. The device relations are chosen from those in Table 2. The valid possible input parameter trajectories are denoted by a list of consisting of pair $(snum, quadrant)$ where *snum* is the serial number from Table 4 and *quadrant* is as specified earlier. The list of lumped parameters is a list of symbols where each symbol is a literal that denotes a lumped parameter such as `Area`, `Gear-ratio` etc. Our representation has a well-defined vocabulary of commonly used lumped parameters that capture device geometry and material properties. Since each case is a physically realizable instance of a device, the lumped structural parameters have fixed values. Many devices that are available allow a range of values for their structural parameters. For example a gear transmission (an assembly of gears) provides a range of gear ratios. For such devices with variable structural parameters, a range of nominal values for the structural parameters are listed. The directed-graph representation of the device topology is an incidence matrix representation wherein each node corresponds to a component denoted by a symbol. The right-hand side of each equation in device, structural and parameter relations in Tables 2,3 and 4 are represented as trees where each node of the tree is the function name (a literal) and the leaves are arguments (literals) of that function. Such trees can be represented as recursive lists in prefix notation. Shown in Figure 9 is the tree representation of the relation $y = m_1.t + m_2.t^2 + c_6$ from Table 4 and its recursive form as a list is `(+ (* m1 t) (* m2 t) c5)`. A complete equation is represented as a pair *(left-hand-side right-hand-side)* wherein *left-hand-side* is a literal naming the parameter and *right-hand-side* is the recursive definition of the function. A device, structural or parameter relation is represented as a list of such equations. Table 8 shows the schema for a DC motor with field current strength variable between two and five amperes.

An organized case-base provides for efficient retrieval during the synthesis process. The cases in the case-base can be classified into a typology as shown in Figure 10. Each level

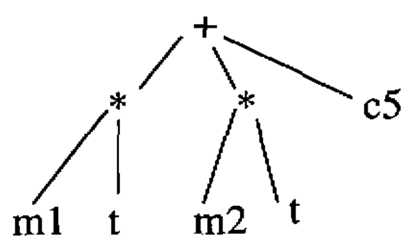

Figure 9: Representation of parabolic parameter relation

Attribute	Value
Device name	Symbolic name of device
Input port	A symbol naming port
Output port	A symbol naming port
Input port energy domain	One of `MT, MR, TH, EM, HY`
Output port energy domain	One of `MT, MR, TH, EM, HY`
Device relation	D_i from Table 2
Feasible inputs	List of valid input parameter trajectories
Structural relation	S_i from Table 3
Structural parameters	List of lumped parameters of device
Structural parameter values	Range of values for structural parameters
Components	List of components of device
Device topology	A directed graph representation of topology

Table 7: Schema for a device

Attribute	Value
Device name	`DC-motor-1`
Input port	`P1`
Output port	`P2`
Input port energy domain	`EM`
Output port energy domain	`MR`
Device relation	$e_2 = k.f_1, f_2 = e_1/k$
Feasible inputs	`((1,+)(2,+)(3,+)(4,+)(5,+)(1,-)(2,-)(3,-)(4,-)(5,-)`
Structural relation	$k_1 = p_1$
Structural parameters	`(Field-current)`
Structural parameter values	`((2 5))`
Components	() (A null list denotes no components)
Device topology	()

Table 8: Schema for a particular DC-motor

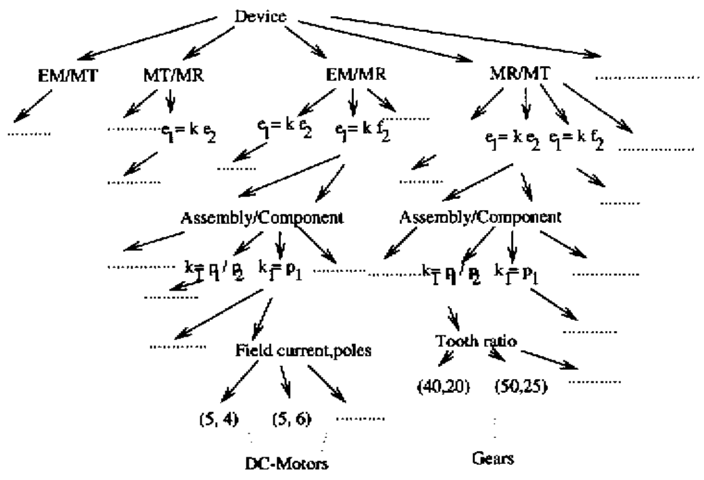

Figure 10: Typology of two-port power devices

Key	Description	Value for DC-motor
k_1	Input and output port energy domains	`EM-MR`
k_2	Device relation	`D3`
k_3	Assembly or component	`Component`
k_4	Structural relation	`S1`
k_5	Structural parameters of device	`Field-current`
k_6	Values of structural parameters	`2`

Table 9: Indexing keys for a device

of the hierarchy refers to a particular attribute of a case. The first-level denotes the energy domains for input and output ports. The second level denotes the device relation. The third level denotes if a case is a component or an assembly. The fourth level identifies the type of structural relation, the fifth level names the lumped parameters and the sixth level specifies the nominal ranges for the lumped parameters of a device. DC-motors and gears are shown indexed by the typology.

The hierarchical classification scheme provides a unique index for every case. A *composite* key, K_i of the type $(k_1, k_2, k_3, k_4, k_5, k_6)$ can be generated and assigned for every case based on the above defined hierarchy. Each k_i that constitutes the composite key is described in Table 9. The first column names the key, the second column describes the key and the third column gives the value of the key for the DC-motor case of the previous example. Thus given that the value of k_1 is `EM-MR` where `EM` denotes that the input port of the case is electro-mechanical and `MR` denotes that the output port of the device is rotary mechanical, we can retrieve the DC-motor case. One can also retrieve DC-motors if it is given that we require all devices that have device relation D_3. It is obvious that the slot values of the case-schema can be used to generate the composite key for a given case. Retrieval of cases from the case-base is performed by specifying the composite key K. The composite key (`EM-MR,D3,Component,S1,Field-current,2`) will retrieve only the specific case in Table 8. A composite key (`EM-MR,D3,Component,S1,Field-current,*`) where * denotes a don't-care will retrieve all DC-motors with field-current as the structural parameter. The key (`EM-MR,*,*,*,*,*`) will retrieve all motors (including AC-motors) and further other assemblies such as motor and gear combinations. The key (`EM-MR,D3,*,*,*,*`) will retrieve all DC-motors only. The attributes that constitute the composite key K are ordered from the most general to the most specific key based on the typology. Since every case in the case-base is indexed by the composite key, the retrieval algorithm uses a given composite key and retrieves all cases that match the key. An *exact match* is required between the corresponding elements of the key to a case and the given specifications in the query. Details of index organization and implementation are beyond the scope of this paper.

Consider the design specification for the gate-drive system in Table 6. The initial design specification only provides element k_1 of the composite key, ($k_1 =$ `EM-MT`). The other elements of the composite key are not specified. *The synthesis task involves generating possible values for those other elements of the composite key that are unspecified.* The parameter trajectories provided in the design specification provide the requisite information to generate the values for other elements of the composite key. Thus design can be viewed as the process of generating all the elements of the composite key and once all the elements are known, a feasible design is obtained. Each design alternative can be viewed as a choice of values for each unknown element of the composite key. Conceptual design is essentially the task of generating values for the elements $k_i, i = 1,2,3,4,5$ and parametric design involves choosing the value for element k_6. As the design process proceeds from conceptual design to parametric design, values for specific keys are identified and the number of feasible cases is further reduced. In the following section, we describe the synthesis algorithm that proposes a variety of possible values for each element of the composite key in a principled manner, retrieves cases based on the proposed composite keys, eliminates infeasible combinations of by evaluating combinations of cases using the parameter trajectory information provided in the design specifications and thus further refines the values of the elements of the composite key to generate valid designs.

3. The case-based design procedure

As described in the previous section, conceptual and parametric design tasks are equivalent to generation of the right combination of values for the elements of a composite key. The design task is complicated since there are *a number of alternatives for each element* of the composite key. Therefore there a number of alternative combinations of these composite key element values that can meet the design specification and thus there are a number of design solutions. We note that we have not provided any subjective criteria such as cost, weight, volume etc. as part of the design specifications. If such information were provided with the design specifications, the additional specifications would be used to choose amongst the variety of physically realizable alternatives.The synthesis process can be organized into two stages,

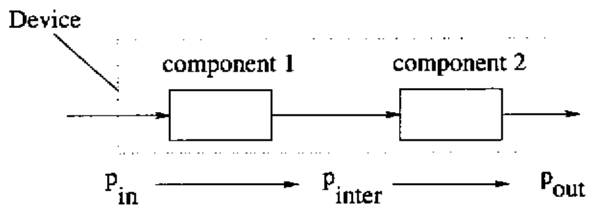

Figure 11: Power-flow path representation in devices

namely, (1) Given a design specification as in Table 6, retrieve cases from the case-base that can satisfy the specification, i.e. transform the input parameter trajectories into the required output parameter trajectories. and (2) If no single case can be found to satisfy the design specifications, then *compose* cases from the case-base to generate a new design. We describe a synthesis algorithm wherein both these tasks are interleaved. The algorithm consists of three essential procedures, namely, ***elaboration***, ***retrieval*** and ***verification***. Elaboration can be viewed as the task of generating all possible alternatives for elements k_1 and k_2 of the composite key. Retrieval is the task of retrieving cases based on values of k_1 and k_2 to obtain possible values for $k_i, i = 3,4,5,6$. If values for more keys are given, then more specific cases would be retrieved. Verification is the task of eliminating all the infeasible cases retrieved by the composite key using the parameter trajectory information provided in the design specification. We describe each of these procedures and present a synthesis algorithm using these basic steps.

3.1. Elaboration

Power flow path from the input port to output port in a two-port device, whose device topology is not known, can be described by the graph $p_{in} \rightarrow p_{out}$, where p_{in} and p_{out} denote input and output power ports. Power flow in a device with two components is given by the graph $p_{in} \rightarrow p_{inter} \rightarrow p_{out}$ where p_{inter} is the common port shared by the two components as shown in Figure 11. The directed graph denotes that the input port of the first component is connected to the environment and the output port of the first component *feeds into* to the input port of the second component. The output port of the second component is connected to the environment. The port p_{inter} is the output port for one component and an input port for the adjacent component. Each arc in the directed graph is equivalent to a component and thus two nodes (ports) in the directed graph are spanned by a component. *Elaboration* is the process of *generating* a directed graph $p_{in} \rightarrow p_1.....p_n \rightarrow p_{out}$ from $p_{in} \rightarrow p_1.....p_{n-1} \rightarrow p_{out}$. Thus elaboration is the process of introducing a new common port in a given power flow path thereby introducing another additional component in the device topology. The elaboration process is ***equivalent to the process of generating an internal structure or device topology*** for a two-port device to transform the input parameter trajectories to output parameter trajectories. Each component that spans two ports provides a particular kind of power transformation depending on the device relation, D_i that relates the the input and output ports of the component. The design specification does ***not explicitly*** provide information on the kind of transformation required by specifying the overall D_i required

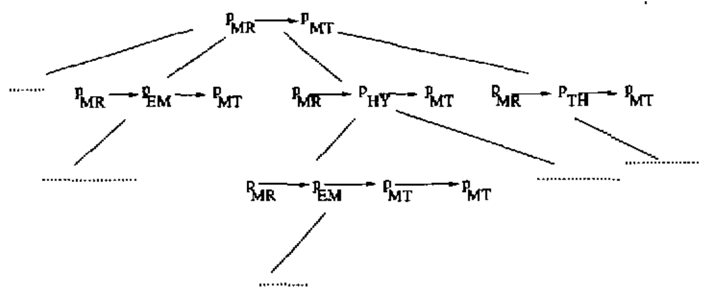
$p_{MR} \rightarrow p_{MT}$
$p_{MR} \rightarrow p_{EM} \rightarrow p_{MT}$
$p_{MR} \rightarrow p_{HY} \rightarrow p_{MT}$
$p_{MR} \rightarrow p_{TH} \rightarrow p_{MT}$
$p_{MR} \rightarrow p_{EM} \rightarrow p_{MT} \rightarrow p_{MT}$

Figure 12: Elaboration tree

between the input and output ports of the required device. This information is *implicit* in the parameter trajectory specifications provided as part of the design specifications. The elaboration procedure proposes possible combinations of component device relations that can meet the specifications.

Elaboration is a heuristic procedure and has been used for case index generation in [10] . Given a directed graph for elaboration, the elaboration procedure splices the arcs between two nodes of the graph and introduces a new node. Elaboration of $p_{in} \rightarrow p_{out}$ generates $p_{in} \rightarrow p_{inter} \rightarrow p_{out}$. The energy domains of the input and output ports of a device are given by the design specification. The intermediate ports that are introduced can belong to one of five energy domains listed earlier. A new port is introduced between the output port and the penultimate port in the directed graph to ensure that only unique directed graphs are generated. Thus if energy-domains of the power ports were considered, introduction of a single intermediate port in a directed graph generates five new elaborated graphs. Thus the elaboration procedure creates a tree of "elaborations" where each node of the tree is a directed graph denoting a particular topology of power flow as shown in Figure 12. In the figure, the input energy domain is mechanical rotation and the output energy domain is mechanical translation. Each node of the tree has an additional port with respect to its parents. Each parent node has five children nodes though all five have not been shown due to space limitations. The directed-graph data structure at each node of the elaboration tree is called an *Elaboration index*. Each elaboration index at every node of the elaboration tree is equivalent to the elaboration index at the root of the tree. For example, a device that provides a power flow path from the electro-mechanical domain (EM) to rotary mechanical (MR) domain is physically equivalent to an assembly of two components that are connected in series, wherein the first one provides a power flow path from electro-mechanical domain (EM) to rotary mechanical (MR) domain and feeds into a component that provides a power flow path from rotary mechanical (MR) domain to rotary mechanical (MR) domain. Hence each elaboration index is a possible alternative for the element k_1 of the composite key to the case that meets the design specifications.

Synthesis of two port devices involves connecting components in a certain topology to meet the input-output trajectory specifications. Elaboration generates all possible device topologies for a two-port device, that are comprised of only two-port devices, given the energy domains of its input and output ports. To identify the components that span two nodes in the device topology, we also need to know the device relation between the two ports. Each arc in the device topology can be described by a device relation from Table 2. Thus if there are m arcs in an elaboration index and N device relations, there are N^m combinations of devices that have the same elaboration index. Each such combination of component device relations is a viable alternative value for element k_2 of the composite case that satisfies design specifications. Since the overall device relation is *not* explicitly provided with the design specifications, we propose all possible combinations of component device relations as viable solutions. Thus each arc of a given elaboration index is instantiated with a device relation to generate a *retrieval index*. A retrieval index with n arcs is denoted by the n-tuple,$(c_1, ...c_n)$ where each c_i is a 5-tuple $(p_{in}, p_{out}, e_{in}, e_{out}, dev_i)$. p_{in} and p_{out} are literals that denote input port and output ports. e_{in} and e_{out} are one of the literals EM,MT,MR,HY,TH denoting the energy domains of p_{in} and p_{out}. dev_i denotes the serial number of the device relation from Table 2 that spans the ports p_{in} and p_{out}.

A composite key of the type (k1,k2,*,*,*,*) can be generated for every arc of the retrieval index wherein k1 denotes the energy domains of the two ports that are at either nodes of the arc and k2 denotes the device relation. Let arc i of the retrieval index retrieve m_i cases from the case-base. Thus a retrieval index with n arcs will retrieve $\prod_{i=1}^{n} m_i$ combination of cases. Consider one combination of cases retrieved using a key-index of n-elements i.e. retrieving one case each for an elaboration index with specified device relations for its arcs. The combination is denoted as a list, $(case_1, ..., case_n)$ wherein $case_i$ is the case schema. Each such combination of cases is called an assembly, A. The assembly, A, will be a singleton set when cases are retrieved for the elaboration index at the root of the elaboration tree. Each assembly that is composed has to be validated to ensure that meets the design requirements. In the following section, we provide a scheme to eliminate all invalid combinations of cases.

3.2. Case verification

Once an assembly is obtained, we need to verify that the assembly can produce the requisite output parameter trajectories for given inputs. An assembly can fail if (1) the device relation of the overall assembly is invalid and (2) if the structural parameters of its component cases are invalid. The overall device relation of an assembly may be invalid when the device-relation for one of its components is invalid i.e. a wrong combination of component devices has been generated. Even if the combination of components can provide the requisite transformation between input and output, they might be invalid because of scaling errors due to erroneous combination of structural parameter values. We present a verification procedure to check whether a given combination of cases can transform the input parameter trajectories to the output parameter trajectories for a single time region. We motivate the procedure with a simple example of validating a single device (case). Assume that a device with device relation, D_1 ,which is $(e_2 = k.e_1, f_2 = f_1/k)$, has been retrieved. For the sake of clarity, we also assume that the retrieved case has the same input-output energy domains as required by the design specifications. The design specifications for the input and output parameter trajectories for the time region are as shown in Table 10. The first column specifies the

Parameter	Specification	Relation
$Pspec_e^{in}$	`((1,0,k1),+,(0 t-end))`	$y = m_1.t + c_1$
$Pspec_f^{in}$	`((1,0, g),+,(0 t-end))`	$y = m_1.t + c_1$
$Pspec_e^{out}$	`((2,0,m),+,(0 t-end))`	$y = m_1.Sin(t) + c_2$
$Pspec_f^{out}$	`((1,c,0),+,(0 t-end))`	$y = m_1.t + c_1$

Table 10: An example specification

effort and flow parameter at the input and output ports and the second column specifies the required the time history of the efforts and flows. The effort and flow trajectories in the specification at input and output ports are denoted as $Pspec_e^{in}$,$Pspec_f^{in}$,$Pspec_e^{out}$ and $Pspec_f^{out}$. The third column lists the parameter relation required in the specification for illustration. $Pspec_e^{out}$ has a sinusoidal trajectory whereas $Pspec_e^{in}$,$Pspec_f^{in}$ and $Pspec_f^{out}$ are linear in nature. `k1,g,m` and `c` denote numeric constants. `(0,t-end)` denotes the duration of each trajectory.

Device relations are validated by *symbolically* solving a system of equations involving the parameter relations specifying the input and output parameter trajectories and the device relation equations. Given a device relation and the output parameter trajectories, the input parameter trajectories can be obtained by symbolically solving for the inputs. For example consider the relation $y = kx$ wherein k is constant. Given that $x = ft^2$ wherein f is constant and t is the independent variable, one can obtain $y = kft^2$ by eliminating x. Alternatively given $y = kft^2$ and $y = kx$, one can obtain x as a function of the independent variable t. Similarly, given the output parameter trajectory as a function of time, we can obtain the input parameter trajectory as a function of time from the device relation by symbolically eliminating the output parameter. Now if the device relation is to be valid, the input parameter trajectory obtained by elimination must match the input parameter trajectory of the design specification else the device relation is invalid.

The output parameter relations $Pspec_e^{out}$ and $Pspec_f^{out}$ and the device relation D_1 are symbolically solved to obtain the input parameter relations $Psolv_e^{in}$ and $Psolv_f^{in}$ as functions of time. If the solved relations $Psolv_e^{in}$ and $Psolv_f^{in}$ match $Pspec_e^{in}$ and $Pspec_f^{in}$, then the device relation is considered valid. Symbolically solving the output specification in Table 10 and D_1 gives $Psolv_e^{in}$ = `m*Sin(t)/k` where `k` is the algebraic coefficient in the device relation and $Psolv_f^{in}$ = `c*k`. The specification $Pspec_f^{in}$ does not match the solved relation $Psolv_f^{in}$ since it is not a constant but a linear function of time. $Pspec_e^{in}$ does not match $Psolv_f^{in}$ since it is not a sinusoidal function. Hence *Psolv* and *Pspec* do not match and we can conclude that the the case retrieved is not a valid solution.

For an assembly of n cases, where $case_1$ is connected to the input port and $case_n$ is connected to the output port, , we repeat the solve-match procedure starting from the output port and $case_n$ to obtain the output specifications for $case_{n-1}$ and proceed to obtain *Psolv* for the input port. We match *Psolv* with *Pspec* for the input port parameters. If they match the combination of cases is considered a valid design that meets the design specifications.

The above described validation scheme can be improved if we note the following: *Pspec* and *Psolv* might not match for two reasons, namely, (1) The two relations might be completely different i.e. they are two different equations and (2) The two equations may have the same morphology but have different coefficients c_i and m_i. Thus matching of *Psolv* and *Pspec* can be done at two levels, (1) where one matches only on the morphology of the relations neglecting the values of the coefficients and (2) where one considers the values of the coefficients too. Matching at level 1 is concerned only with the classes of device relations while at level 2 we are concerned with the structural parameter values. If matching at level 1 fails then it means that combination of cases retrieved by the retrieval index will not result in a valid assembly that can would have the desired behavior. Further details of matching parameter relations are presented in Section 4. The elaborate-retrieve-solve cycle is repeated for each time region of the parameter trajectories. Assemblies (combination of cases) that can handle the complete trajectory are finally returned as viable solutions to the design specification. In the following section we present the complete algorithm and illustrate it with solutions generated for the gate-drive system.

3.3. Synthesis algorithm

The main steps of the synthesis procedure are as follows:

1. The design specifications for the first time interval, T_1, are chosen and an elaboration tree is created. The root index of the tree has the energy-domains as in the design specifications.

2. Device relations are introduced in the elaboration index and retrieval is performed with the retrieval indices thus generated. Case retrieval returns all cases that match the given energy domains and device relation.

3. For each case thus retrieved, case verification is performed by symbolically solving the device relation and output time histories for the input time-histories. Cases that are successfully verified are further checked for the values of their structural parameter values.

4. Successful cases are returned as solutions for the first time interval and checked with the design specifications for the remaining time intervals.

5. If no cases are obtained, the root index is further elaborated by adding a new port and thus a new component. Device relations are introduced for each arc of the elaboration index and cases are retrieved for the retrieval indices thus obtained. Cases are verified again by symbolic elimination for each of the time intervals. The procedure iterates until a successful solution is obtained or a predetermined level in the elaboration tree is reached.

We illustrate each of the above steps of the procedure with the synthesis of a device topology for the gate-drive system.

3.4. Synthesis of the gate drive system

1. The design specifications for the gate-drive system are shown in Table 6. The effort and flow trajectories have been discretised into three regions and the synthesis procedure begins with the initial time region set to the opening half-cycle of the gate which is the first time-region. An elaboration tree has the root elaboration index $p_{EM} \rightarrow p_{MT}$ is instantiated.

2. Retrieval is performed with retrieval indices $p_{EM} \xrightarrow{D_i} p_{MT}$ where each D_i is a device relation from Table 2. A key, K, generated from the retrieval index $p_{EM} \xrightarrow{D_1} p_{MT}$ will be (`EM-MT,D1,*,*,*,*`). Case retrieval is performed with these indices.

3. In the case-base implemented, there are no power devices that directly span this energy domain. Hence no cases are retrieved for any retrieval index for the elaboration index $p_{EM} \rightarrow p_{MT}$. Hence further elaboration is required.

4. Elaboration of the index $p_{EM} \rightarrow p_{MT}$ generates five new indices. Elaboration indices $p_{EM} \rightarrow p_{EM} \rightarrow p_{MT}$ and $p_{EM} \rightarrow p_{MT} \rightarrow p_{MT}$ are not considered since a sub-index ($p_{EM} \rightarrow p_{MT}$) of that index has failed. Retrieval indices are generated for the elaboration index $p_{EM} \rightarrow p_{MR} \rightarrow p_{MT}$. The combination of cases for the elaboration index $p_{EM} \rightarrow p_{MR} \rightarrow p_{MT}$ are listed in Table 11. The first column gives the device relations for each arc (from input to output) and the second column lists the corresponding devices. Table 12 lists the device relations in terms of the effort and flow relationships for convenience. The retrieval index $p_{EM} \xrightarrow{D_3} p_{MR} \xrightarrow{D_1} p_{MT}$ retrieves the combination of cases (`Induction-motor, Rack-pinion-mechanism`). The retrieval index $p_{EM} \xrightarrow{D_3} p_{MR} \xrightarrow{D_1} p_{MT}$ retrieves the combination of cases (`DC-motor, Rack-pinion-mechanism`). The sub-index $p_{MR} \xrightarrow{D_5} p_{MT}$ retrieves a slider-crank mechanism, scotch-yoke and sinusoidal cam mechanism. The sub-index $p_{MR} \xrightarrow{D_1} p_{MT}$ also retrieves a straight-line cam mechanism.

5. Case verification is performed for each possible combination of cases shown in Table 11. Consider the assembly consisting of the `AC-motor` and `Rack-pinion` mechanism. Solving $e_2 = k.e_1, f_2 = f_1/k$ and the output specifications $e_{out} = 500$ and $f_{out} = 0.1$ we obtain that the torque (effort) output of the AC-motor must be $500/k$ and the angular velocity (flow) output must be $0.1 * k$. k is the tooth-ratio of the rack-pinion mechanism. Since cases are instantiated for structural parameters in the case-base, k will have a value and we assume a case with k= 20 has been retrieved. Hence the torque required is 25 Nm and the angular velocity 2 rad/sec. We solve $e_2 = k.f_1.Sin(\mathcal{F}_2), f_2 = e_1/k.Sin(\mathcal{F}_2)$ for a motor with $k = 5$, to obtain that the input voltage to the motor must be $10Sin(16)$ and input current must be $5Cosec(16)$. We compare this with

Device relations	Cases
D_3,D_1	`(DC-motor Rack-pinion),(DC-motor Straight-line-cam)` `(DC-motor Linear-screw-mechanism)`
D_3,D_5	`(DC-motor Slider-crank-mechanism),(DC-motor scotch-yoke)` `(DC-motor sinusoidal-cam)`
D_9,D_1	`(AC-motor Rack-pinion),(AC-motor Straight-line-cam)` `(AC-motor Linear-screw-mechanism)`
D_9,D_5	`(AC-motor Slider-crank-mechanism),(AC-motor scotch-yoke)` `(AC-motor sinusoidal-cam)`

Table 11: Cases retrieved for index $p_{EM} \rightarrow p_{MR} \rightarrow p_{MT}$

No	Relations
D_1	$e_2 = k.e_1, f_2 = f_1/k$
D_3	$e_2 = k.f_1, f_2 = e_1/k$
D_5	$e_2 = e_1/(k.Sin(\mathcal{F}_1)), f_2 = f_1.k.Sin(\mathcal{F}_1)$
D_9	$e_2 = k.f_1.Sin(\mathcal{F}_2), f_2 = e_1/k.Sin(\mathcal{F}_2)$

Table 12: Device relations for cases in Table 11

the design specifications and find that a constant, non-sinusoidal voltage is the input specification. Thus all combination of cases with an `AC-motor` will be rendered invalid.

Consider the design **`DC-motor`** and **`Rack-pinion-mechanism`**. Considering the same rack-pinion mechanism as retrieved above, the torque output requirements for the motor is 25 Nm and the angular velocity is 2 rad/sec. We solve $e_2 = k.f_1, f_2 = e_1/k$ for the `DC-motor` with $k = 10$, to obtain input voltage required is 20 volts and an input current of 2.5 amperes. The voltage and current parameter relations, obtained by solving, match the specifications since both are constant values. Voltages are constant and since the relation of electrical current as function of time is not specified, any relation is considered valid. Thus a `DC-motor` and `Rack-pinion` combination is a viable solution. Though the voltage relations match morphologically, their values are not equal. The synthesis algorithm considers the above combination as a failure and checks all different combinations of `DC-motors` and `rack-pinion` mechanisms that exist in the case-base till the correct combination is located. If no such combination exists in the case-base it fails and proceeds to try another retrieval index. We have not implemented procedures that hypothesize structural parameter values once a morphological match is obtained.

6. A feasible solution is a `DC-motor` with motor constant $k = 25$ and a `rack-pinion` mechanism with a gear-ratio of 20. We note that an infinite number of combinations are possible parametrically. The algorithm retrieves only those that are available in the case-base. The case base serves as a source of devices that have been physically

realized and work. Hence, retrieving devices from the case-base eliminates possible downstream failures due to manufacturing and eliminates searches for other parameter combinations. Other possible solutions for the opening half cycle are `DC-motor` and `Straight-line-cam` and the combination of `DC-motor` and `Linear-screw-mechanism`. Composition of device relations (D_3,D_5) fails to produce any solution because of the sinusoidal term in the device relation D_5 and case verification results in failure.

7. The feasible solutions are verified for each of the remaining two time regions. The search procedure terminates once solutions are found and case retrieval is not performed for the elaboration indices, $p_{EM} \rightarrow p_{TH} \rightarrow p_{MT}$ and $p_{EM} \rightarrow p_{HY} \rightarrow p_{MT}$. Valid combination of devices as solutions for the gate drive system are the combinations: `(DC-motor Rack-pinion)`,`(DC-motor Straight-line-cam)` and `(DC-motor Linear-screw-mechanism)`.

In the following sections, we analyze the complexity of the search and discuss heuristics that have been implemented to guide the search.

3.5. Complexity of the synthesis procedure

Computational efficiencies for retrieval algorithms are primarily determined by the organization of the case-base. The hierarchical classification and generation of indexing keys provides a near linear performance. The exploration of the design space is determined by the branching factor of the elaboration tree and the number of device relations that are defined. At a given level n of the elaboration tree,(level 0 is root,) there are $n+1$ arcs. Let there be totally m possible device relations. Also if the case-base has N instances for each device relation, then the total number of designs that need to be explored at level n is at most $(Nm)^{n+1}$. The branching factor of the elaboration tree is five since we consider only five energy domains. Thus the total number of combinations of cases searched to a depth n is at most $\sum_{i=0}^{n} 5^i (Nm)^{i+1}$. Thus the search performed is affected both by the number of cases in the case-base and the number of device relations in Table 2. The domain heuristics described in the following guide the exploration of this very large space. The search efficiency can be improved as the system acquires more cases. The acquired assemblies can be directly retrieved, thus reducing the number of combinations of individual components explored. For example, if assemblies of motors and gears were represented as cases in the system, the elaboration index $p_{EM} \rightarrow p_{MR}$ will retrieve not only motors but also motor-gear assemblies that were generated by the elaboration index $p_{EM} \rightarrow p_{MR} \rightarrow p_{MR}$ thus reducing search. The synthesis procedure can also be improved by caching often used components and assemblies and thus speeding up the retrieval process.

4. Domain heuristics for elaboration and matching

The algorithm presented in the previous section uses a variety of heuristics to focus the search for cases. Domain heuristics that capture knowledge regarding device relations, nature of inputs and outputs and device topologies are used to guide the search to find solutions efficiently. In the worst case, the elaboration tree search is *exponential* in nature. In the following subsections, we present heuristics that guide elaboration, retrieval and matching in the synthesis algorithm.

4.1. Elaboration and retrieval heuristics

Elaboration indices in the elaboration tree are searched *breadth-first* to obtain the *smallest number* of combination of components in a design. An elaboration-index can fail to retrieve cases either when there are no devices that span the requisite energy domains or the combination of devices that spans the required energy domains fails the case-verification procedure. With the exception of the root index, if no cases are retrieved for a particular elaboration index (called E_{fail}) in the elaboration tree due to the first possible reason, further exploration of elaboration indices of which E_{fail} is a sub-index is pruned. If no cases were retrieved for the elaboration index $p_{MT} \rightarrow p_{MR} \rightarrow p_{MR}$, search will be aborted for the elaboration index $p_{MT} \rightarrow p_{MT} \rightarrow p_{MR} \rightarrow p_{MR} \rightarrow p_{MR}$.

An important stage in the search process is the choice of an elaboration index, a node in the elaboration tree for performing case retrieval. Elaboration indices whose intermediate ports have the *same energy domains as the input or output ports* are prefered to indices that have a variety of mixed energy domains. The reasoning behind this heuristic is that devices built from components belonging to same energy domains are easier to build, test and control. Thus an elaboration index $p_{MT} \rightarrow p_{MR} \rightarrow p_{MR}$ will be prefered to the index $p_{MT} \rightarrow p_{HY} \rightarrow p_{MR}$.

Case retrieval is based on the retrieval indices generated from an elaboration index. Retrieval indices that have linear device relations between ports are prefered to retrieval indices with sinusoidal and other non-linear relations. Key indices with linear device relations are used to access the cases before other combinations of device relations. This is to ensure that simpler combinations of devices that are easier to control are generated before more complex combinations.

4.2. Heuristics for symbolic solving and matching parameter relations

Symbolic solving of equations is a critical step in this synthesis procedure and is used as a mechanism for verifying the combination of cases. Solving and matching input-output effort and flow relations with time provides a *robust* mechanism for case verification. This verification scheme is robust in the sense that there are no ad hoc validation rules. It is considerably general to handle a large variety of functions of time as inputs and outputs to devices. The procedure also enables verification of parameter trajectories at both the morphological and parametric levels.

Symbolic equation solving critically depends on the nature of the (1) device relations and (2) input-output parameter relations. In this section, we present the symbolic solving schemes for different types of device relations and parameter relations. Discussion of symbolic solving algorithms per se are beyond the scope of this paper [1]. The solving procedure also uses the following rules for verification of input and output parameter relations:

- It is not possible for a device to have null input parameter relations and non-null output parameter relations. This captures the fact that without any input no outputs can be produced.
- At either input or output port, if the effort parameter relation is null, then the flow parameter has to be null and vice versa. This is the constraint that to supply power to a system you need to have both effort and flow parameter as non-null entities.
- By convention, both effort and flow parameter relations must belong to the same quadrant. This is to enforce the constraint that power is a scalar and is always positive.

Symbolically solving for a parameter relation only gives information about the form of the parameter relation as a function of time. No information is provided regarding the quadrant of the parameter relation. The rule is that the quadrant of the input parameter relations is the same as the output parameter relations unless the device relation has a sinusoidal term involving an integral variable in it or the structural relation imposes a negative algebraic coefficient of the device relation. A sinusoidal term in the device relation can change the quadrant of the output only if the value of the integral variable is greater than an odd multiple of π. For device relations involving non-sinusoidals, there is no effect of the integral variables in terms of relating input-output parameter relation quadrants. The length of the interval of discretisation obtained after solving is equal to the length of the interval of the input parameter relations. The symbolic solving scheme also performs the necessary integration required for device relations involving effort and flow integral terms. During symbolic solving, for each case, the procedure also checks if the input parameter relations to the device are valid since the valid relations are defined in the case schema.

Matching solved relation, *Psolv*, and the design specification relation, *Pspec* is performed in two stages. Two parameter relations match if both their patterns are same i.e. they match morphologically. Since parameter relations are represented as trees, two patterns are considered to match if the nodes and leaves of the tree match each other. A match is obtained at the device relation level, if the numerical values at the leaves of the trees are considered as unconstrained constants and the literals at the leaves are of the same type i.e. effort or flow parameters. The literals at the nodes of the tree must exactly match each

[1] We use Mathematica[22] as a back-end to the search mechanism to solve equations.

other since they are names of mathematical functions. Two relations match parametrically if the numerical values at the leaves of their tree representations are equal.

5. Discussion and concluding remarks

The synthesis procedure has been implemented as part of the CADET system [10, 8]. This approach enhances the influence graph based synthesis scheme by providing a convenient scheme to link device structure and behavior. In this section, we discuss the advantages and limitations of this synthesis methdology.

The features of the proposed device model and case representation are:

- The bond graph based device model captures energy interactions between devices. The notion of entities such as components and assemblies are well-defined as also the notion of *assembling* two components i.e. connecting devices at their ports. This scheme is well-suited for reasoning about device dynamic behavior.
- The device model provides for integrating both conceptual and parametric design in a coherent manner. Also the notion of device relations, structural relations and parametric relations aid in modeling a wide variety of device behaviors both in terms of their input-output relation and effort-flow variations with time at the input and output ports.
- From a case-based reasoning point of view, each case is a unique device which is an instantiation of a prototypical device defined by the device relation. The device representation provides the device and structural relations as a set of discriminatory indices for retrieving cases. The representation also provides for a convenient scheme for classifying design cases from the dynamic behavior perspective.

The synthesis algorithm has some limitations. At present, transient behavior of devices is not addressed. The device models capture *ideal* energy behaviors. Also only steady-state dynamic behavior of devices has been considered. In the proposed model, spatial orientation of components has not been represented and hence configuration design tasks are not supported. Also the device models do not enable reasoning about geometry and form. Device topologies are considered to be *open-loop* wherein there are no energy or signal feedback loops from components downstream of the power-flow path to components upstream. *Closed-loop* systems involve *signal* feedback from components downstream to components upstream. The synthesis algorithm generates only open-loop systems. Though the synthesis procedure can identify the open-loop components in a closed-loop system, it cannot identify and retrieve signal components that are used to sense and transmit signals. Thus in solutions to the gate-drive system, the algorithm cannot completely generate designs 3 and 4 in Figure 2. Future work aims to extend the capability of the algorithm to identify closed-loop components. Extension to multiple-input multiple output power systems is possible by allowing

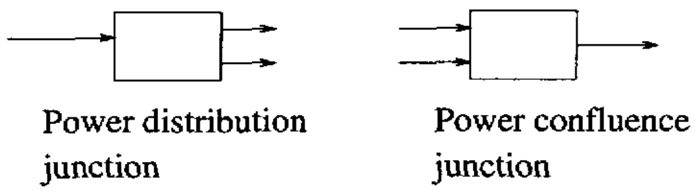

Figure 13: Junction structures in elaboration

for introduction of junctions by the elaboration procedure. Junction structures as those shown in Figure 13 can be introduced to allow for multiple power flow paths. A distribution junction provides for power distribution and a confluent junction for power accumulation.

We have described a CBR-based algorithm for synthesis of single-input single output power drive devices based on bond graph device models. The algorithm combines both conceptual and parametric design tasks. The synthesis algorithm uses design information regarding both device topology and device behavior. Future research aims to extend the synthesis procedure for multiple input and output power drive systems and also consider components that exhibit energy storage and dissipation behaviors.

References

[1] G. Pahl and W. Beitz. *Engineering Design.* Springer-Verlag, London, 1988 edition, 1988.

[2] S.Finger and J.R. Dixon. A review of research in mechanical engineering design. *Research in Engineering Design*, 1(1), 1989.

[3] B. Williams. *Invention from First Principles via Topologies of Interaction.* PhD thesis, Massachusetts Institute of Technology, 1989.

[4] K.P. Sycara, Navin Chandra D., S. Narasimhan. Qualitative reasoning methods in design. In A. Kusiak, editor, *Intelligent Design and Manufacturing.* Wiley Inter-science, 1992.

[5] A. Goel. *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving.* PhD thesis, Dept. of Computer and Information Science, Ohio State University, 1989.

[6] S. Kota. A qualitative matrix representation scheme for the conceptual design of mechanisms. In *Proc. of the ASME Design Automation Conference (21st Biannual ASME Mechanisms Conference)*, 1990.

[7] Srikanth M. Kannapan and Kurt M. Marshek. Design synthetic reasoning: A methodology for mechanical design. *Research in Engineering Design.* To be published.Locate journal issue and volume.

[8] D. Navin Chandra, R. Guttal, J. Koning, S. Narasimhan, Sycara, K. Cadet: A case-based synthesis tool for engineering design. *International Journal of Expert Systems*, 4(2):157–188, 1992.

[9] J Kolodner. *Case-based Reasoning.* Morgan Kaufmann Publishers Inc., San Mateo,CA,USA, 1993.

[10] K. Sycara S. Narasimhan Navin Chandra, D. A transformational approach to case based synthesis. *AI EDAM*, 5(1), 1991.

[11] A. Goel and B. Chandrasekaran. Integrating model-based reasoning and case based reasoning for design problem solving. In *AAAI-88 Design Workshop*, St. Paul, MN., Expected in 1989.

[12] K. T. Ulrich and W. P. Seering. Synthesis of schematic descriptions in mechanical design. *Research in Engineering Design*, 1(1), 1989.

[13] J. R. Rinderle Finger, S. A transformational approach to mechanical design using a bond graph grammer. In *Proceedings of the First ASME International Design Theory and Methodology Conference*, September 1989.

[14] S. Hoover and J.R. Rinderle. A synthesis strategy for mechanical devices. *Research in Engineering Design*, 1:88–103, 1987.

[15] J.R. Rinderle and L. Balasubramaniam. Automated modeling to support design. In *Second ASME Design Theory and Methodlogy Conference.* ASME, 1990.

[16] R. Welch and J.R. Dixon. Conceptual design of mechanical systems. In *Design Theory and Methodology*, volume 31, pages 61–68, 1991.

[17] W. Gosling. *The design of engineering systems.* Wiley, 1962.

[18] T.N.Madhusudan. A review of bond graph based design methodologies. Technical Report CMU-RI-TR-95-28, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA 15213, July 1995.

[19] D. Karnopp and R. Rosenberg. *Analysis and Simulation of Multiport Systems.* The M.I.T. Press, 1968.

[20] D. Karnopp and R. Rosenberg. *System Dynamics: A Unified Approach.* John Wiley Sons, New York, 1975.

[21] R. Rosenberg and Karnopp. D. *Introduction to Physical System Dynamics.* McGraw-Hill Book Co., 1983.

[22] S Wolfram. *Mathematica, a System for Doing Mathematics by Computer.* Addison Wesley, second edition, 1991.