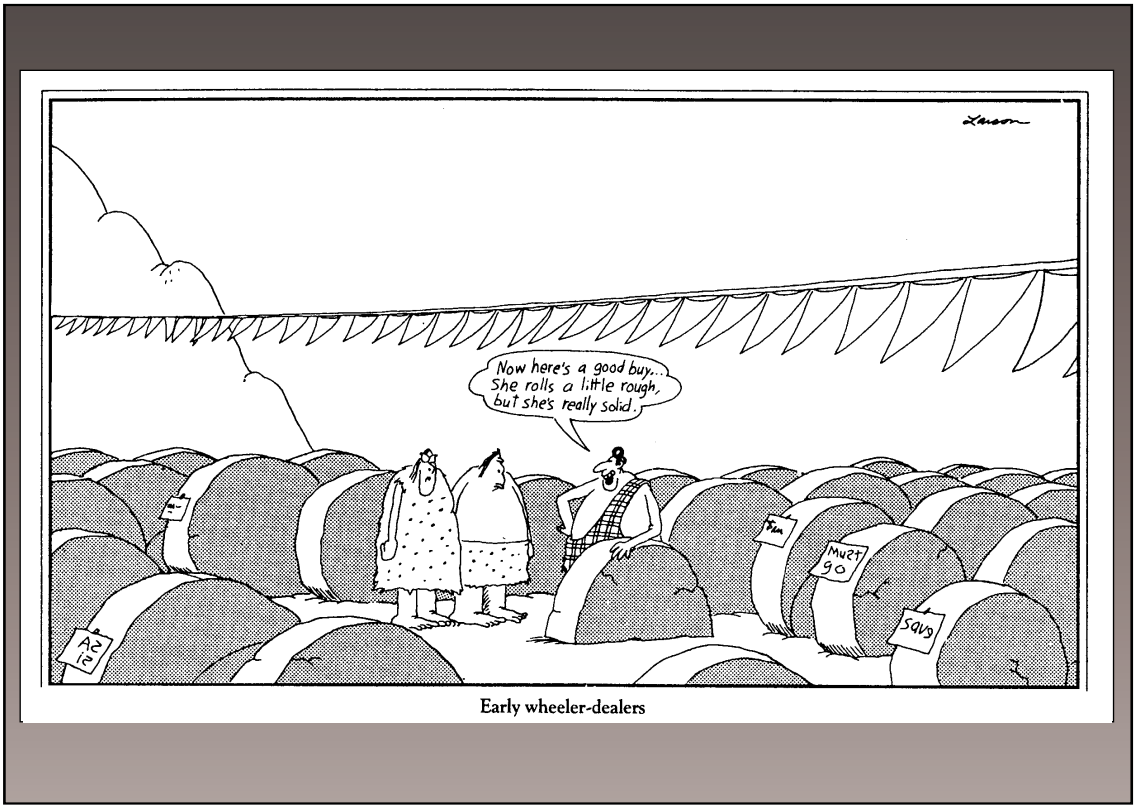
Part I

Representing Mechanical Devices

There are three reasons for building computer representations of mechanical devices: (1) identify the types of structures used, so as to extend the types of inference that can be applied to device-related applications, (2) identify structures that globally organize the problem, so as to control knowledge access and inferencing, and (3) find structures that focus attention on the functionally salient elements of a problem, so that suitable plans and devices can be identified or constructed. In FONM, the structures are associated with a physical object and its function, because an object is used for a reason (or purpose), to achieve some end, and an object function is applied and produces specific behavior. Function organizes how objects are viewed by those who use them and by how objects behave regardless of how they are viewed.

Representing mechanical problem-solving requires access to diverse types of knowledge, and to their interactions, on many levels. The central problems for any computational system are how information is represented, organized, accessed, and manipulated. This part of the dissertation addresses the first three problems, while Part 2 addresses the fourth point. Frame based approaches to knowledge representation ([Minsky 1975], [Schank and Abelson, 1977]) construct patterns as abstract knowledge templates that declaratively represent clusters of concepts, variables, and relations. As information is obtained, the patterns are instantiated and are then used to provide inferences.



Early wheeler-dealers

Chapter 2
Device Statics

Devices are used to perform problem solving tasks. Devices can range from simple geometric shapes to complex organizations of interacting subsystems. The description of a device at rest, its construction and appearance, is important in determining whether it can support a particular task. The device's *construction* refers to the composition and connectivity of its components. The device's *appearance* refers to the shape and size of its components and their features, and must support the device's behavior and remind the problem solver of the task. There are four elements of a device static description: the composition of its components, the appearance of its components, the orientations of its components, and the connectivity of its components. Device function is dependent on the orientations between and connectivity of its components. Device use is dependent on the functions the object can effect and also on how the object appears to the problem solver. Object composition, appearance, orientation, and connectivity are aspects of the static device description, and their interaction is called *device statics*.

In this chapter, the representation constructs which are used in FONM to describe device statics are presented. Every device and device component can be described as a collection of individual objects, so the discussion begins with a description of individual objects and their physical characteristics. The discussion ends with the presentation of static representations for fifteen mechanical devices of increasing complexity. Five topics are addressed in this chapter:

(1) Physical objects and individuals
(2) Geometric primitives
(3) Object primitives in FONM
(4) Relational characteristics and connectivity between objects
(5) Device static representations

2.1 Physical Objects and Individuals

Regardless of its overall complexity, the components of every device can be recursively decomposed to a level of individual objects. An *individual object* is defined as a set of physical characteristics associated with an object type. A stick, or rock, is an individual object, because to describe it further requires a description of its material properties, its size, and its shape.

2.1.1 Object Physical Characteristics

In FONM, an object has two types of physical characteristics: its *properties* and its *regions*. Object properties are divided into those related to the object's material composition, which determine how the object can behave under various conditions, and those related to me-

chanical perturbations, which determine what behavior has happened or can happen. Object properties are represented with physical states. Object regions describe general locations on the object by their shape, and size, and determine how and where the object can be connected and manipulated.

Physical States

An object can be represented as a collection of physical states, each of which describes a different property and its value. Physical states are represented using a knowledge structure with four roles: obj, prop, dimr and val. The *obj* role refers to the physical object whose property is being represented, the *prop* role is associated with a symbolic property name, the *dimr* role is associated with the applicable dimension and direction for which the property value is valid, and the *val* role is associated with the knowledge structure representing the quantity associated with the property value. In order to represent physical states, these roles and their fillers must be described.

Properties

Every object has a finite number of physical properties. In FONM, four material properties are used to represent object composition:

(1) material stiffness
(2) material density
(3) material strength
(4) material opacity

Material stiffness is a measure of whether an object can transmit mechanical force to other objects. Density is a measure of whether an object can restrain another object's motion. Strength is a measure of whether and how an object will absorb energy. Opacity is a measure of whether an object can transmit sensory information.

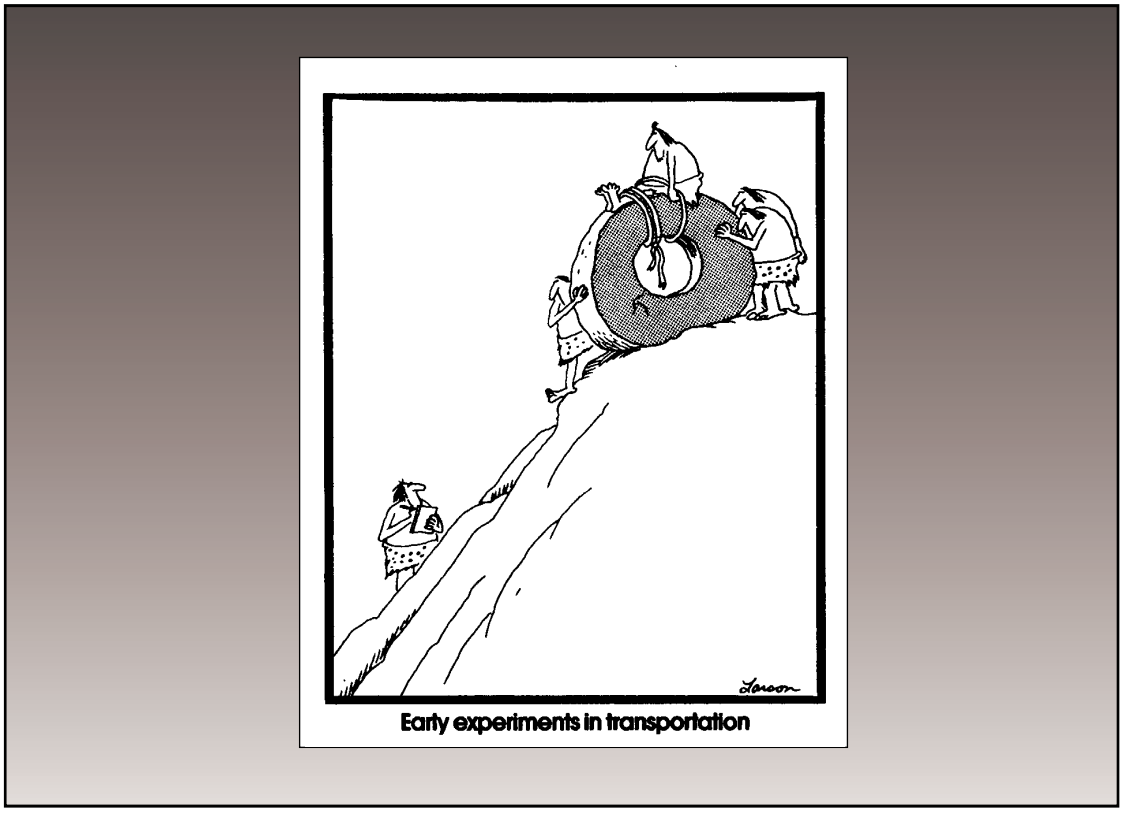
In addition, five properties are used to represent the types of mechanical behavior an object can exhibit:

(1) position
(2) restraint
(3) applied force
(4) internal force
(5) size

Position is associated with object motion. *Restraint* is associated with object connectivity. *Applied force* is associated with motion and force transmission. *Internal force* is associated with energy storage. *Size* (and shape) is associated with deformation.

Dimension and Direction

Each object property has an applicable dimension and direction. In FONM, dimension and direction are combined in a state's *dimr* role. The values associated with dimension are those for the cartesian axes (x, y, z) and their cylindrical equivalents (r, z), as shown in Table 2.1**:**



Chapter 4
High-Level Dynamics and Device Function

In this chapter, the representation constructs which are used in FONM to describe high-level device dynamics are presented. Every device and device function can be described in terms of a finite set of objects which have associated with them a specific function. These *machine primitives* comprise a set of functional building blocks for representing complex devices. The functions of these primitives are represented with behavioral sequences, so the discussion begins by defining a structure for function which is based on device behavior, and which is based on the physical requirements of objects which can instantiate them. The discussion ends with the presentation of dynamic representations for six devices of increasing complexity. Three topics are addressed in this chapter:

(1) Device function
(2) Machine primitives (MPs)
(3) MP combinations and mechanical device representation

4.1 Device Function

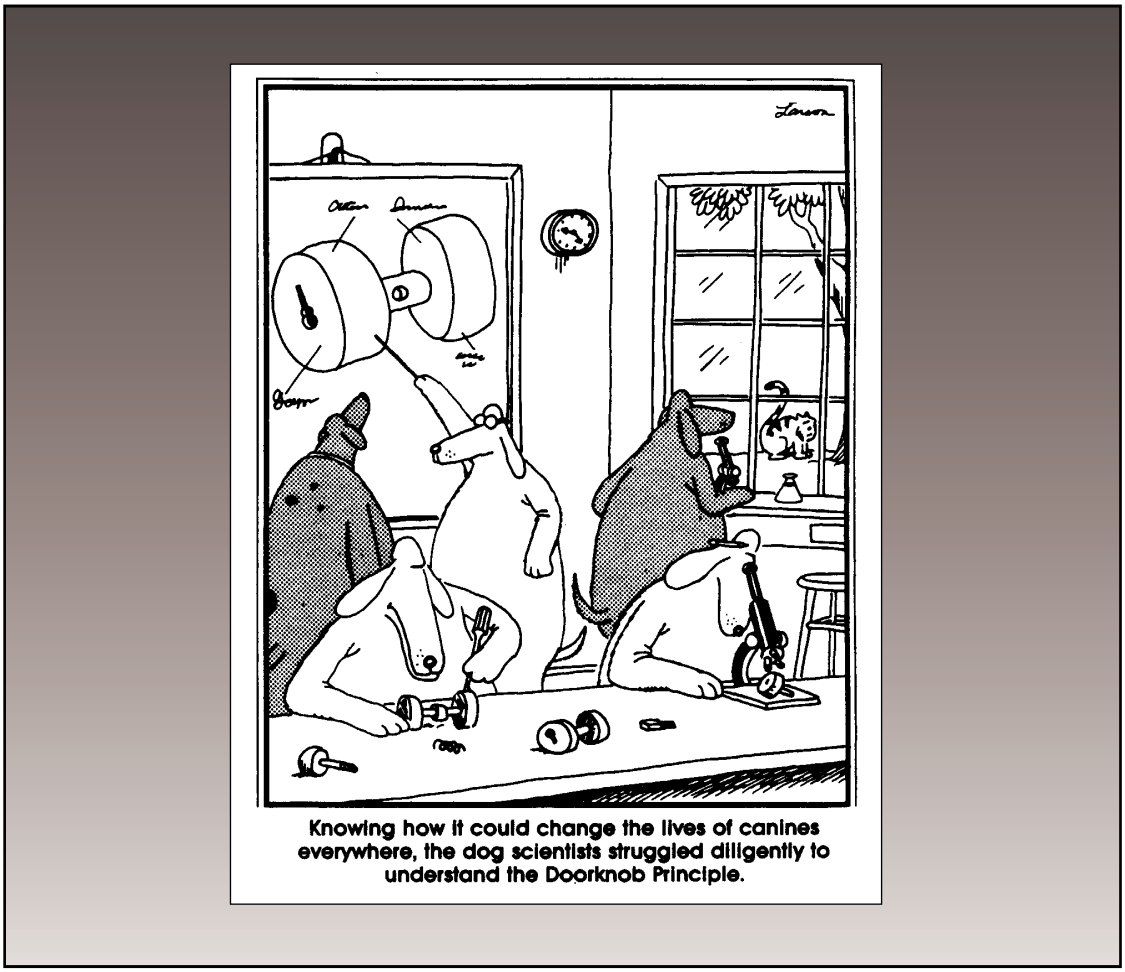
A *function* describes *what* a device does and is invoked by perturbing a device. For example, two boards can be connected with screws using a screwdriver. The function of the screwdriver is to provide the necessary force needed to drive the screw into the boards. The driving function is invoked by turning the screwdriver when it is in contact with a screw, and results in a magnification of the rotational force applied at the handle to the screw head. A device may instantiate different functions, depending on what force is applied, to what region of the device the force is applied, and the device statics. The screwdriver can be used to punch holes in oil cans, or to pry staples from documents. Driving, punching, and prying are each invoked by applying a different type of force to the same object. Each function makes use of different aspects of the device statics, and each produces a different result. In each case, the overall function of the device can be represented as a combination of the functions of its components, and they, in turn, can be represented as BPP sequences.

The shaded portions of Figs. 3.31 - 3.33 represent the underlying behavioral sequence of the screwdriver components. If the shaded boxes are darkened, so that the details of the component behavior are not visible, as in Fig. 4.1, then the boxes represent the individual functions of the screwdriver components in the overall driving function. The only information entering the boxes are the enabling states from the applied force and from the boxes on the right, which represent the connectivity relations between the components (device statics). The only information leaving the boxes are the resulting states. This is called a *black box* description because it describes what happens when the device is perturbed, but not how it happens.

PART II

Reasoning About Mechanical Devices

The overall goal of the FONM representation theory is to support the design and implementation of computational models of mechanical reasoning, particularly improvisation and invention. As a creative problem-solving task, invention can be viewed as an integration of four reasoning tasks: (1) problem recognition, (2) problem interpretation, (3) problem evaluation, and (4) experimentation. This part of the dissertation presents two computational models constructed while developing FONM. Each model makes use of specific aspects of the FONM representation of devices, and demonstrates recognition of knowledge and inference-making capability in its reasoning domain.






Chapter 6
Mechanical Experimentation

This chapter presents a program which performs mechanical simulation. EDEXP (EDison EXPerimenter) is a computer program that takes the symbolic representation of a simple mechanical door, mutates the door by moving a hinge, simulates the capacity of the door to move, by moving the hinge to different locations, and identifies the relationship between hinge behavior and door rotation. To accomplish this task, EDEXP implements the FONM theory of MOTION and RESTRAIN behavioral primitives.

In this chapter, the reasoning processes associated with hypothesizing, executing, and interpreting the results of mechanical experiments are discussed with respect to their dependency on how knowledge of mechanical behavior is represented and manipulated. The EDEXP model is presented and applied to a door manipulation example.

6.1 Computational Architecture

The two models presented in this and the next chapter utilize the computational architecture from the EDISON project, but have been implemented using different versions of FONM and different computational approaches. As such, those components of the architecture which are specific to each program are presented with respect to the kind of I/O and behavior that the program produces.

6.1.1 EDEXP

EDEXP implements four components of the EDISON computational architecture (Fig. 6.1): (1) a planning demon interpreter, (2) rules for performing planning analysis, mutation, and recognition (3) a planning memory, and (4) a long term memory of abstract patterns

associated with devices, device plans, and device behaviors.

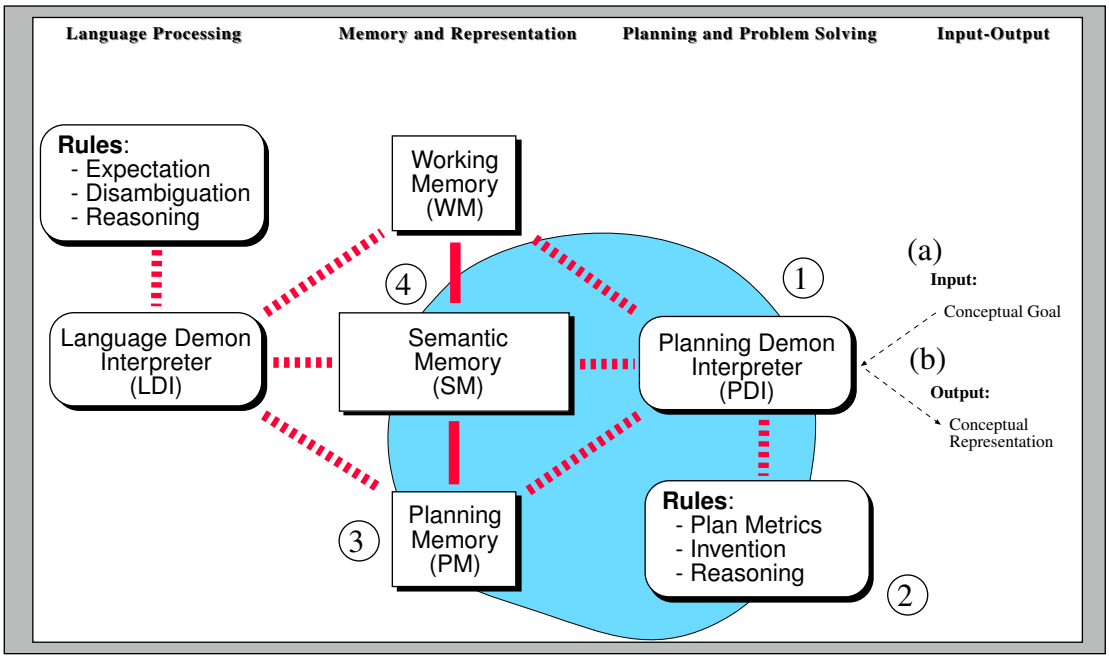


Figure 6.1 The components of EDEXP. Thick solid lines represent inter-memory access. Thick dotted lines represent control access to memories and rules from demon interpreters. Thin dotted lines represent input/ output data flow.

- **Semantic Memory (SM)**. Semantic memory represents schematic knowledge of objects, events, and their causal interactions. In EDEXP, SM is comprised of frame-based rules which describe object behavior according to the FONM behavioral process definitions. During planning and behavioral simulation, input conceptualizations are compared to their counterparts in semantic memory, which are used to instantiate incomplete frames.

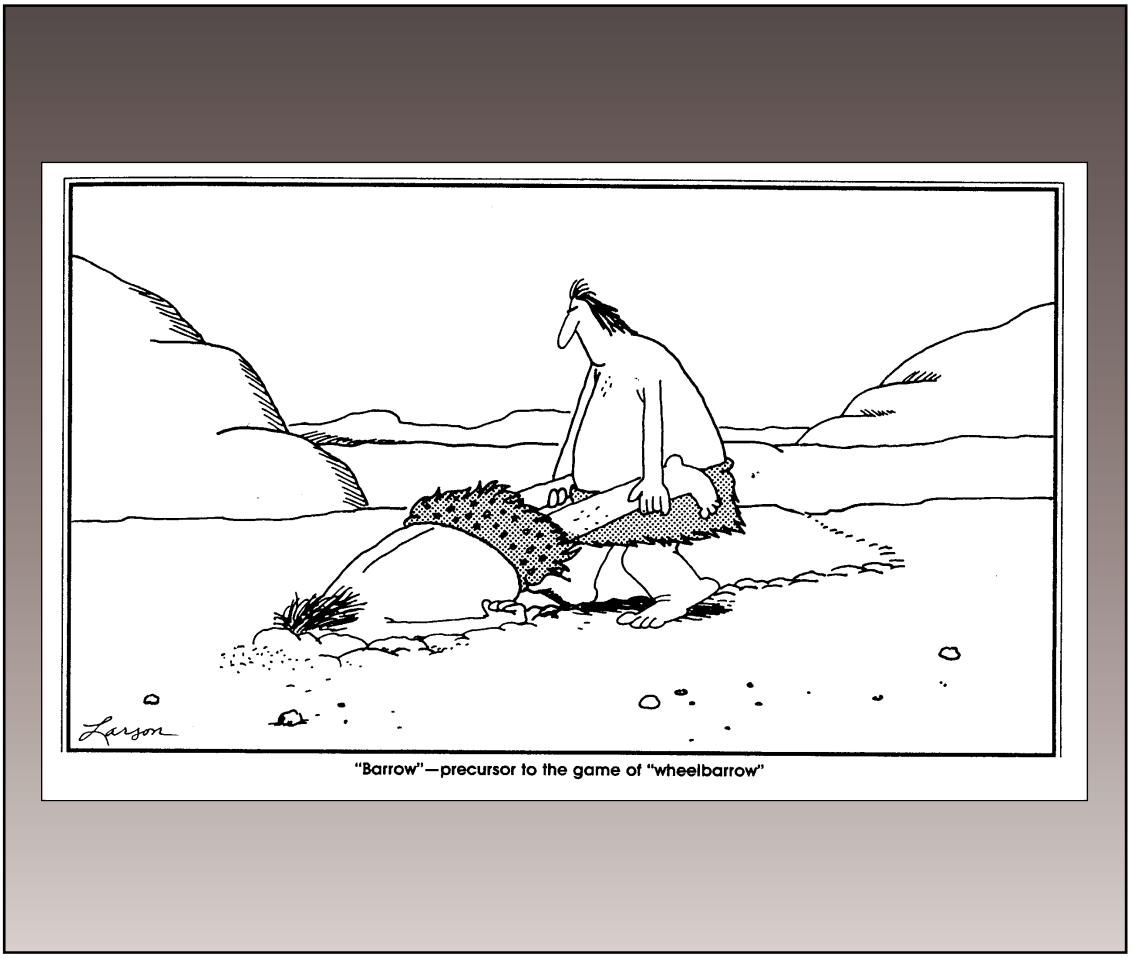
- **Planning Demon Interpreter (PDI)**. The planning interpreter uses demons to process goals and plans by recursively matching their subgoals to planning memory and comparing the subgoals to the overall task. The interpreter functions in a two modes: (1) experimentation, and (2) hypothesis testing. During experimentation mode, the plans associated with the experiment and device are instantiated with the intent of generalizing a hypothesis which relates the goal and device. During hypothesis testing mode, the planner applies the hypothesis to the chosen plan and device templates to attempt to prove the hypothesis.

- **Planning Memory (PM)**. During planning, EDEXP uses instantiated conceptual frames initially placed onto a memory called *planning memory*. Planning memory serves as a memory retrieval source for reasoning since the items accessed from memory represent the episodic input. Planning memory is maintained until a plan is analyzed.

PART III
Related Work, Scope, Extensions and Conclusions

The value associated with the FONM representation theory is determined as a function of how it relates to existing work, how well it addresses the problems it is intended to address, and to what extent it provides new knowledge in a rapidly evolving discipline. This part of the dissertation consists of three chapters which address the value of FONM.



"Barrow"—precursor to the game of "wheelbarrow"

Chapter 8
Comparison to Related Work

This chapter is organized in three sections as follows: First, the models and systems that were used as a starting point for the design and construction of FONM are briefly discussed. FONM is then compared to AI representation approaches that describe devices at the abstraction levels addressed by FONM. Finally, the FONM demonstration models are compared to AI systems that are similar in scope.

8.1 Background Work Supporting FONM and its Models

The approach used to develop FONM makes extensive use of Roger Schank's contributions to knowledge representation, that of (1) modeling naturally occurring language tasks and behavior, and (2) knowledge representation and organization in terms of sets of atomic primitives. The work of Schank and his students has been utilized in the following two areas.

- **Conceptual structures**: The Conceptual Dependency representation of actions [Schank 1973, Schank 1975], the script representation of event sequences [Schank and Abelson, 1977, Cullingford 1978], the representation of named plans, and the goal type taxonomy [Schank and Abelson, 1977].

- **Demon-based control**: Using demons and agendas to organize and control process knowledge as implemented in BORIS [Dyer 1983], IRON-FINDER [Reeves 1989], and THUNDER [Reeves 1991], and also by Lenat in AM and EURISKO [Lenat 1976, Lenat 1983, Lenat and Brown, 1984].

8.1.1 Device-Use

The design of FONM device-use plans is an extension of the Schank and Abelson USE(object) plan. In Conceptual Dependency theory, two methods are used to represent action sequences: scripts and plans. A *script* is a stylized template which describes prototypical or compiled action sequences. A *plan* is an abstract template which describes the method, or sequence of actions, as a collection of subgoals. Scripts are very stylized, and are recognized, and processed, quickly. The design-intended use of a device is readily recognized when the device is mentioned, and is very stylized in its application. A script has two drawbacks with respect to the representation of device use. First, if the device fails to work, for one reason or another, a script does not support recovery, so it is not a suitable structure for problem solving or creativity. Second, scripts represent known applications, so they cannot be used to represent unintended device functions. Plans are not processed quickly but are more general than scripts. Because of its general structure, if a portion of a plan fails, the entire plan doesn't automatically fail. Unlike a script, both an explanation for the failed por-

tion, and an alternative method for recovering from the failed portion, can be suggested. This is more in line with machine fault diagnosis and problem solving.

Schank and Abelson proposed two structures for addressing the strengths and weaknesses of scripts and plans when applied to device use: (1) named plans, and (2) instrumental scripts. The *named plan* was proposed as an intermediary between a script and a plan, and has characteristics of both. The named plan describes routine activities, and is represented as "a fixed sequence of subgoals which form the usual path to goal attainment." The *instrumental script* was proposed to represent "fixed and uninteresting actions" associated with device application, such as driving a car or working a keypunch, and was specifically associated with named plans [Schank and Abelson, 1977]. Being a fixed sequence, the named plan is scriptlike. Being represented as a sequence of subgoals, the named plan is planlike.

8.1.2 Object Use

Schank and Abelson's USE(object) is a named plan used to represent the application of objects toward the achievement of some goal. In their notation, USE(object) represents a sequence of four subgoals and an action, the template for which is shown in Eqn. (8-1):

$$\text{USE(object)} = \text{FIND(object)} + \text{D-CONT(object)} + \text{I-PREP(object)} + \text{DO} \qquad (8\text{-}1)$$

Each component on the right hand sides of Eqn. (8-1) and Eqn. (8-2), except DO, represents a subgoal to be achieved on the way to enabling the USE(object) plan. Finding an object means knowing that the object is appropriate, knowing where it is located, and getting there. Schank and Abelson defined the FIND(object) subgoal as follows:

$$\text{FIND(object)} = \text{D-KNOW(loc(object))} + \text{D-PROX(loc(object))} \qquad (8\text{-}2)$$

where the object may be a name of a specific object or a template for an object with a specific property. The DO in Eqn. (8-1) represents the action taken by the actor once the subgoals have been achieved. The addition signs (+) denote a conjunction between the subgoals and the actor's action. In the Schank and Abelson model, d-goals (e.g., D-CONT) are called delta goals, and i-goals (e.g., I-PREP) are called instrumental goals. D-goals and i-goals support the achievement of higher level goals (i.e., crisis, preservation, satisfaction, achievement, and entertainment) [Schank & Abelson, 1977]. For example, a high level goal to preserve one's health (a P-HEALTH goal) is associated with a number of plans, one of which might be to brush one's teeth. Three delta goals which support the achievement of preserving one's teeth are knowing (D-KNOW) the location of the toothbrush, getting to (D-PROX) that location, and gaining physical control (D-CONT) of the toothbrush. These goals can be achieved a number of ways, such as with the actions of remembering, walking, and grasping, respectively. There is a direct causal relationship between delta goals and the actions, scripts, and plans which are used to achieve them. An instrumental goal is the preparation (I-PREP) of the toothbrush for use. This goal is achieved by locating the toothpaste, opening the container, placing toothpaste onto the toothbrush bristles, and moving the toothbrush to one's mouth. Whereas the delta goal leads to new actions or plans, the instrumental goal generally leads to instrumental script application, where the causal relationship is indirect because the representation level needed to specify how the goal is achieved is beyond the scope of the Schank and Abelson approach.

With respect to device use representation, the USE(object) plan is problematic for four reasons. Consider the USE(object) plan applied to the task of brushing teeth, as illustrated

in Fig. 8.1 The first three subgoals (D-KNOW, D-PROX, and D-CONT) can be satisfied

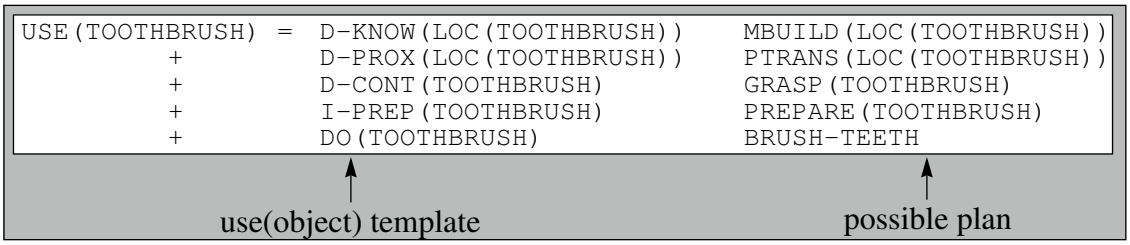


Figure 8.1 A possible instantiation for the USE(TOOTHBRUSH) plan.

with actions such as MBUILD (e.g., “remember”), PTRANS (e.g., "walk"), and GRASP (e.g., "pick up"). The remaining subgoals, however, should be resolved with new actions, scripts, or plans. Any of these which describe device application should be causally related to the device’s function and behavior. None of the Schank primitive acts, instrumental scripts, or plans are formally related to device function or behavior. Second, the preparation of the toothbrush for use involves a number of objects and actions. For example, the application of toothpaste implies a separate USE(TOOTHPASTE) plan. It is difficult to represent PREPARE(TOOTHBRUSH) with an instrumental script with a USE(TOOTHPASTE) plan embedded within it. Third, if the toothpaste container is found to be closed, a problem arises for the representation of PREPARE(TOOTHBRUSH) as an instrumental script, because the script implies that there is one way to prepare the toothbrush for use and does not allow for alternative plans, such as first opening the toothpaste tube. This is a general problem when scripts are underspecified. Finally, the procedure of brushing ones teeth involves complicated robotic movement, none of which is represented at the scriptal level. In order to effectively represent device use, the USE(object) plan must be modified to address these issues.

These considerations motivated the development of device-use plans. The FONM device-use plan extends the USE(object) plan by considering more than one object at a time, and by considering the dependencies between the different objects which are part of the plan. The development of device-use plans resulted in the parallel development of goals specifically associated with the application of device function.

8.2 Related Work in Representing Mechanical Devices

A primary motivation in developing FONM has been to provide an integrated mechanism for representing knowledge about devices at different levels of abstraction. Although many systems have been devised for representing and reasoning about physical systems (e.g., Hayes’ Naive Physics Manifesto [Hayes 1979], DeKleer and Brown’s conduits in ENVISION [DeKleer and Brown, 1983], Chittaro’s Multi Modeling model [Chittaro et al, 1993]), seven approaches share close similarities with the representational scope of FONM: (1) Forbus’ Qualitative Process theory [Forbus 1984], (2) Sembugamoorthy’s and Chandrasekaran’s Functional Representation [Sembugamoorthy and Chandrasekaran, 1986], (3) Rieger’s Common Sense Algorithm [Rieger 1975], (4) Lehnert’s Object Primitives [Lehnert 1978], (5) Lind’s Multilevel Flow model [Lind 1982], (6) Ulrich’s bond-graph model [Ulrich and Seering, 1987], and (7) the Finite Element Method [Clough 1962].

8.2.1 Qualitative Process Theory

Qualitative Process (QP) theory is a low-level, qualitative, approach for representing the behavior of physical systems, particularly thermodynamic systems and heat transfer [For-

bus 1984]. In QP theory, all object behavior is represented with processes. The process is associated with an individual view, or snapshot, of an object, and describes the interactions and influences which define what the object can and will do. QP theory enables object simulations to be performed, because a qualitative process provides the causal mechanism for describing changes in object properties with time, called an object state history. The strength of the QP approach is that it describes the causal nature of object behavior and interaction regardless of setting or intention. The Forbus model emphasizes the role of the process in making inferences about object relationships.

In QP theory, objects are represented as a collection of quantities, a substance, and a location, as shown in Fig. 8.2

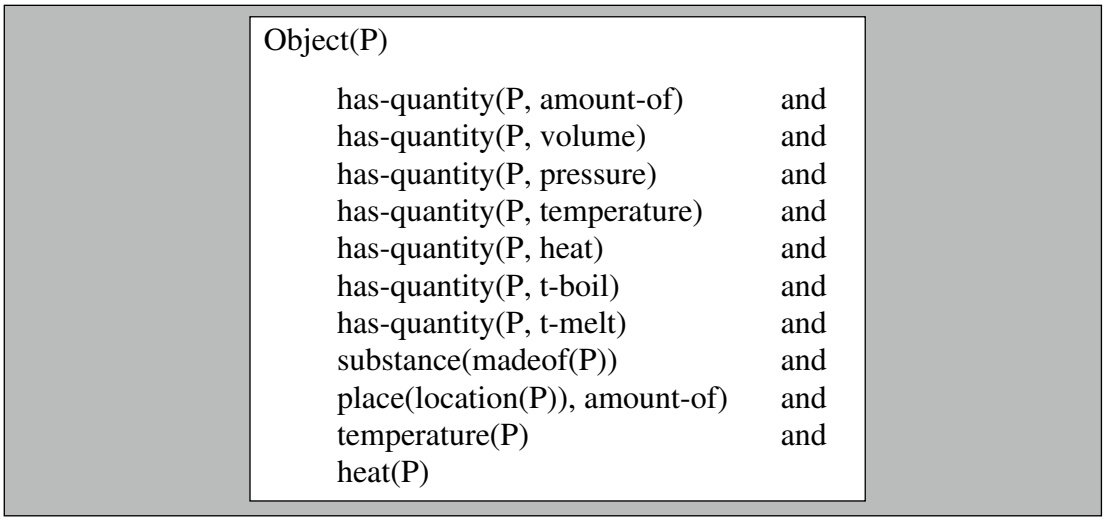


Figure 8.2 Object descriptions in Forbus' QP theory.

Although the quantities which describe objects, in Fig. 8.2, describe thermodynamic properties of objects, any quantities can be represented with this model. Objects can participate in what Forbus calls an *individual view*, which represents a collection of objects and the relations between their quantities. Individual views describe known states which can be

used to recognize behavior, for example for moving friction, as shown in Fig. 8.3.

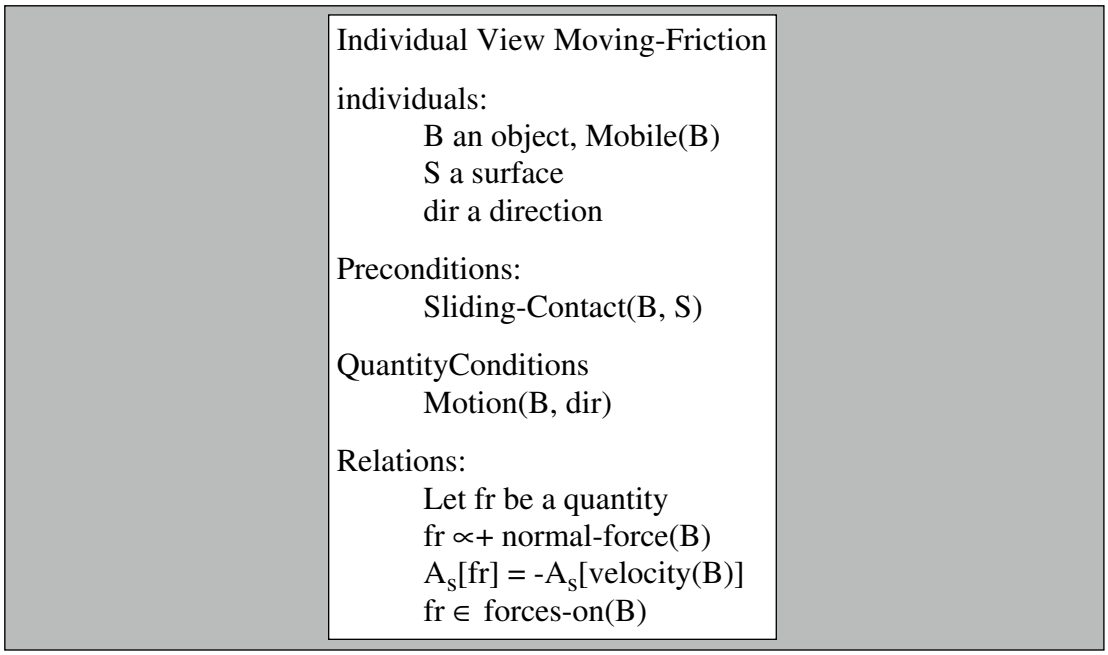


Figure 8.3 Forbus individual view for moving friction.

The individual view in Fig. 8.3 describes a static relation between an object (B) and a surface (S), each of which would be represented by Forbus as shown in Fig. 8.2. Objects which satisfy this definition are in a state of moving friction. The *preconditions* in the individual view describe predicates which must be satisfied by the object descriptions. The *quantity conditions* describe process requirements for the specified objects. In this example, the object (B) must be in motion in direction dir. The *relations* describe interactions between the objects and their quantities while this view is in effect. In this case, a new quantity, friction, is introduced, and its relation to the object's velocity and other forces on the object is defined. QP individual views represent a snapshot of an object in context with other objects and enables recognition of the types of behavior the object is either engaged in or can engage in. In QP theory, processes control the change of object state and individual views. Processes have 5 components: objects, behavioral preconditions, quantity preconditions, behavioral relations between objects, and the resulting influences or tendencies (states). For

example, in Fig. 8.4 the process associated with motion is illustrated.

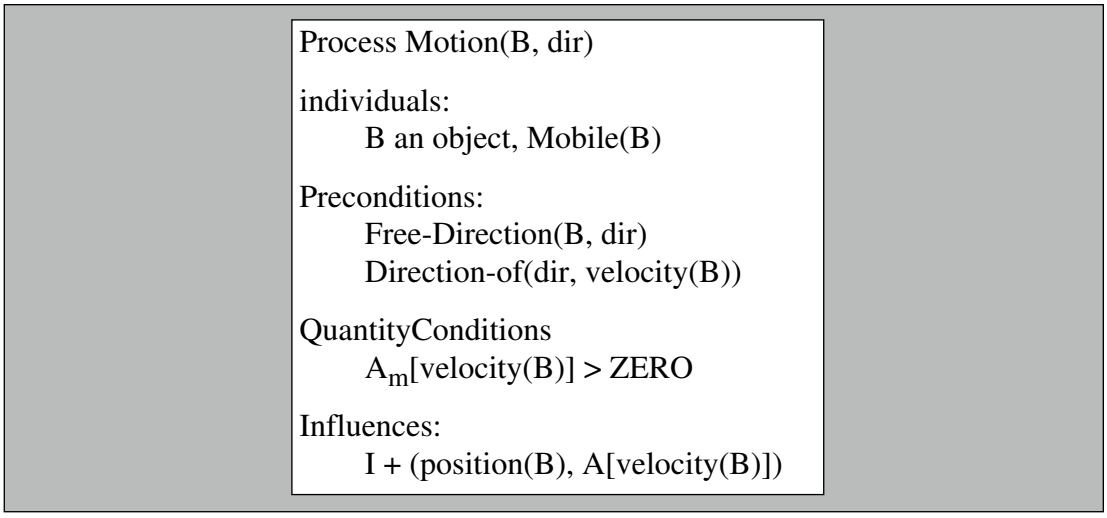


Figure 8.4 Qualitative process for motion.

The difference between an individual view and a process is that the process, through the *influences* component, describes changes to the individual views of objects, and thereby, the ability to generate behavioral histories (i.e., change) for an object. For example, the process for motion depicted in Fig. 8.4 describes a schema for objects which satisfy a mobile predicate in direction dir and have a velocity in dir. The motion influence affects the position and velocity of the object.

Objects in FONM are also represented as a collection of quantities and position. However, instead of a material type, material properties are also represented as quantities. The quantities which can be influenced by processes are distinguished from those which cannot. FONM also describes objects in terms of appearance. QP theory only describes object function in terms of object behaviors, so there is no need for representation of appearance and how it affects function.

The QP and FONM process representations are very similar. Both process models represent the same type and level of information. The FONM process identifies process roles and keeps them separate from the dynamics of the process definition, whereas the QP process combines static and dynamic process components. For example, the FONM process *src* and *dst* roles are represented in a QP process *individuals* role. The FONM process *behavioral preconditions* are equivalent to QP process *behavioral preconditions*. The FONM *perturbation preconditions* are equivalent to the QP *quantity preconditions*. The FONM *from* and *to* roles identify the change of state associated with the behavior, and the QP *influences* role provides a relation for determining the change of state. The algorithmic nature of the QP process influences relations is more detailed than the FONM *from* and *to* states, since it provides the information to calculate state histories.

The closest FONM analog to the notion of the QP *individual view* is a static device representation. The individual view is a construct which helps to recognize an object's behavioral status, and accounts for its dynamic dependencies with other objects and processes. In FONM, the active dynamic processes for an object would be instantiated as a collection of structures, but not as a single structure. The static representation would provide access to those dynamic structures, so some of the same information is represented. For example, the Moving-Friction individual view illustrated in Fig. 8.3 requires recognition that the object B is in motion (not just mobile) and to have access to its velocity for calculating the friction

force value. In FONM, friction is also a force, but would also be associated with a STORE process. As soon as an object is recognized as having surface contact and has having been in motion, the active TRANSFORM process would enable recognition of a STORE process and the effect of friction would be accounted for by distributing it into applied and internal forces (i.e., object motion is reduced because part of the applied force is now 'missing'). In this respect, the QP process and individual view representation has an equivalent FONM representation. By all accounts, the QP representation of friction is a more accurate representation of the behavior, because the relationship between velocity and force is described. In FONM it is only alluded to.

The intended domains of QP theory and Behavioral Process Primitives overlap. BPPs are not intended to support simulation to the same degree of granularity as QP processes, but to enable recognition of when a process is enabled or disabled, and to identify the type of state an enabled process results in. BPPs cannot perform detailed diagnosis, since there is no internal representation for space and time, or for rates of change such as velocity and acceleration. For example, in FONM, when an object stops moving, the object either ceased to be forced or its path was somehow blocked, because these are the two MOTION process enablements.

8.2.2 Device Functional Representation

Sembugamoorthy and Chandrasekaran have developed an approach for representing and reasoning about device function [Sembugamoorthy and Chandrasekaran, 1986]. Their claim is that a device is used to perform a task, and as long as its behavior can be predicted and diagnosed, the internal behavioral relationships can be ignored. The definition of device function is based on representing the function of devices on three levels: (1) causal, (2) temporal, and (3) interaction. The *causal* level identifies function in terms of the behavior which is produced by the correctly functioning device, such as the buzzing of a household buzzer. The *temporal* level describes the temporal dependencies between these behaviors, and the *interactive* level describes how information (material flow, force, etc.) is communicated between components. Within this scope, the *Functional Representation* (FR) recursively describes function in terms of the relations between its components. There are five components to the representation of functional knowledge: (1) the structural relationships between components, such as connectivity, (2) the function, or what the device does as a result of stimulus, (3) the behavior, or how, given a stimulus, the behavioral response is accomplished, (4) generic knowledge, such as rules for particular laws of physics, which contribute to the device behavior, and (5) assumptions which must be satisfied to support the behavior. These components are illustrated below (Fig. 8.5) for the household

buzzer and represented shown represented in Fig. 8.6

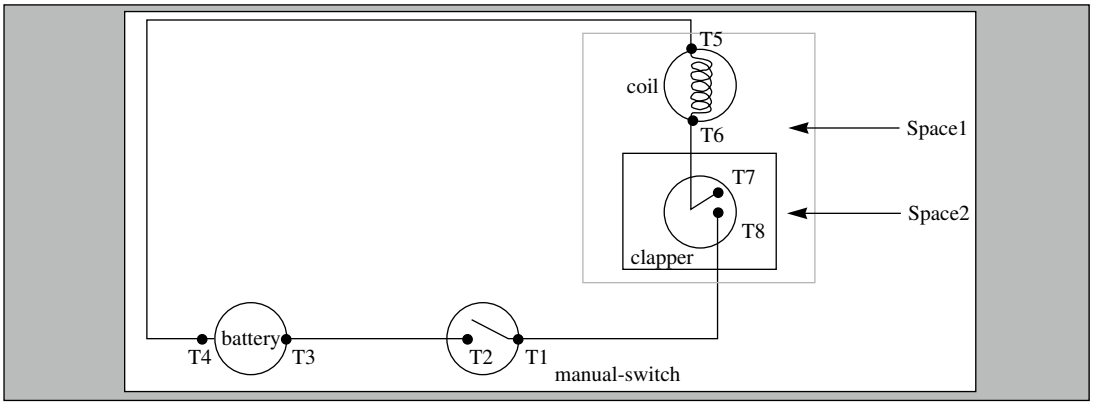


Figure 8.5 A schematic diagram of a household buzzer.

In Fig. 8.5, T1-T8 represent *terminals* of the components, and Space1 and Space2 represent locations in which relations can be effected. FR components of the buzzer representation

in shown in Figs. 8.6, 8.7

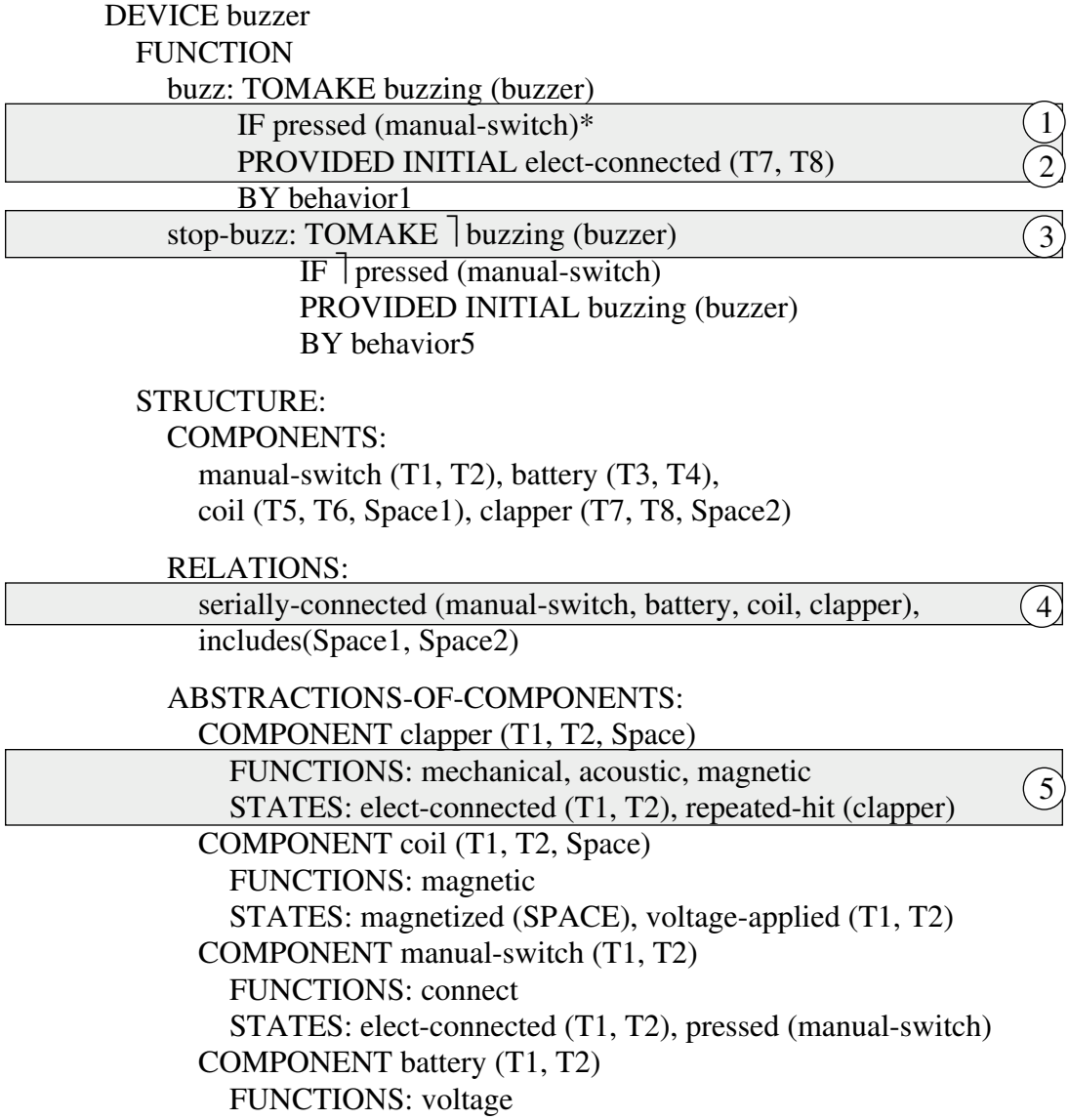


Figure 8.6 FR representation of buzzer.

Fig. 8.6 illustrates the function and structure components of the FR buzzer representation. The function aspect of the representation identifies what happens (TOMAKE buzzing, at 3), how the function is initiated (IF pressed, at 1), and its enablement conditions (PROVIDED INITIAL, at 2). It also identifies how to stop the device from buzzing. A primary assumption of FR is that function is directly associated with purpose. The function TOMAKE thus identifies the purpose of the device and what is produced.

The structure component of the representation identifies the physical components, their relations, and their abstract definitions. Behavioral relations such as *serially-connected* (4) and *elect-connected* (5) are described by the generic knowledge component of the representation. The abstract definition entails identifying affiliated FUNCTIONS and STATES associated with the component. Examples of the FR behavior and generic knowledge com-

ponents of the buzzer representation are shown in Fig. 8.7

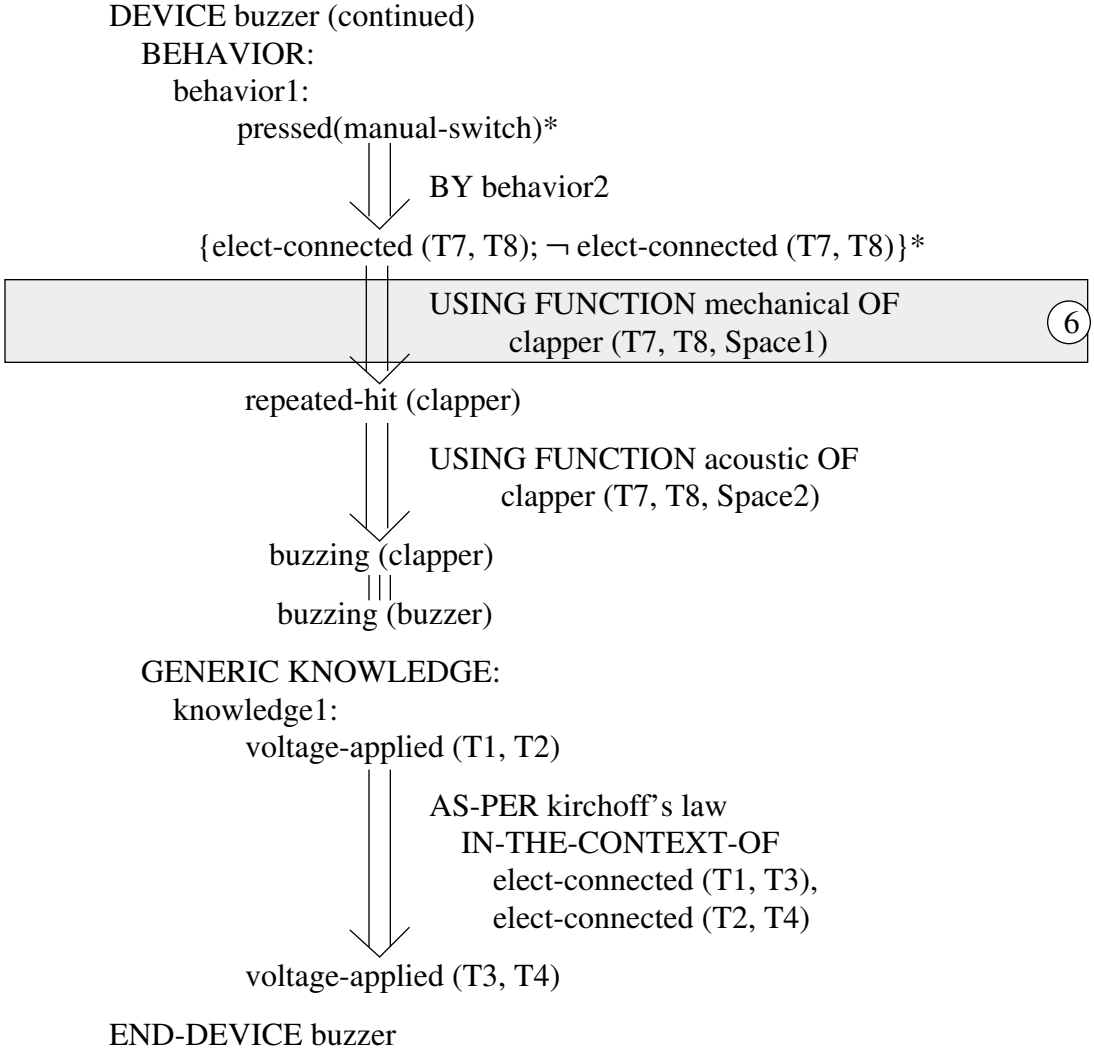


Figure 8.7 FR behavior and generic knowledge representation components of the electric buzzer.

The structure of the FR behavior component for a buzzer indicates that behavior is represented as the functions of the components of the buzzer, rather than their explicit behaviors as would be the case in QP or FONM. For example, in Fig. 8.7 the USING FUNCTION mechanical OF clapper statement (at 6) mediates the electrical connection and the clapping result. The behavior of the clapper: its position, movement, and striking are not specifically described, so the creation of sound is not specifically represented. Since the functions of objects in FR identify their behavioral enablements, the approach can be useful for malfunction diagnosis to the extent of identifying a disfunctional component, but not of explaining why or simulating how.

The FONM approach is basically a functional representation approach, since the interpretation and application of objects, in context, is based primarily on the functional nature of a problem-solver's experience with objects. There are three differences between the models. First, FONM makes the assumption that what an object does (its function), is not one and the same with its purpose but, rather, that a function is never invoked without a purpose. The purpose is itself represented as a goal, and the means whereby the goal is

achieved, by invoking the device function, is represented with a plan. The FONM approach provides access to the goal/plan representation level independently of the device function, behavior, and structural levels and vice versa. Likewise, causal relationships are defined between each of the different knowledge structures, and they can be combined and manipulated independently of one another. The FR device representation captures the same kinds of information but cannot make use of it independently of the device description.

Second, FONM proposes a taxonomy of 11 machine primitives for describing object functions, which help to classify and recognize devices. The Functional Representation provides a representation structure for describing function but, like Qualitative Process Theory for behavior classification, makes no claims for a closed set of device functions. Each device function can be compared along metrics of TO MAKE, etc., but there is no classification of TO MAKE functions which aid in comparing similar devices. The FONM representation for a LEVER function can be applied to comparing any two objects which can instantiate the function and thus potentially be considered as replacements for one another.

Third, the knowledge associated with FR device representations takes the form of rules which are identified with the device. FONM device representations have device-specific rules which are identical in nature to the FR rules. FONM knowledge structures also have rules associated with them which can be used to make inferences about devices which are not specifically associated with a device representation. The FR representation is useful for prediction and diagnosis as long as the intended device function is appropriate to its context. However, no mechanism is provided for relating the device to why it is used, what it was expected to do, or how it was expected to do it outside of the intended context.

In other respects, the FR structure component is equivalent to the FONM static representation. The FR behavioral component is similar to the FONM combination of machine primitives to describe device function in terms of the functions of its components. FR has no equivalent for the behavioral sequences which describe component functions. Nor does FR have any representation for object shape or appearance and their affect on device function and behavior.

8.2.3 CSA and the Representation of Device Function and Application

Rieger's *Common Sense Algorithm* (CSA), integrated Schank's action primitives and object functionality to describe commonsense algorithmic knowledge, or the causal interactions between actors and objects and between objects and other objects [Rieger 1976, Rieger and Grinberg, 1977]. Rieger introduced the notion of an "agentless" action, called a tendency, with the capacity to describe influences such as gravity. He also introduced a family of causal links which were used to describe various causal (temporal) relations and their effects. For example, using CSA, Rieger was able to describe the use of various devic-

es, including a bicycle, a flush toilet, and the bulb bicycle horn detailed in Fig. 8.8.

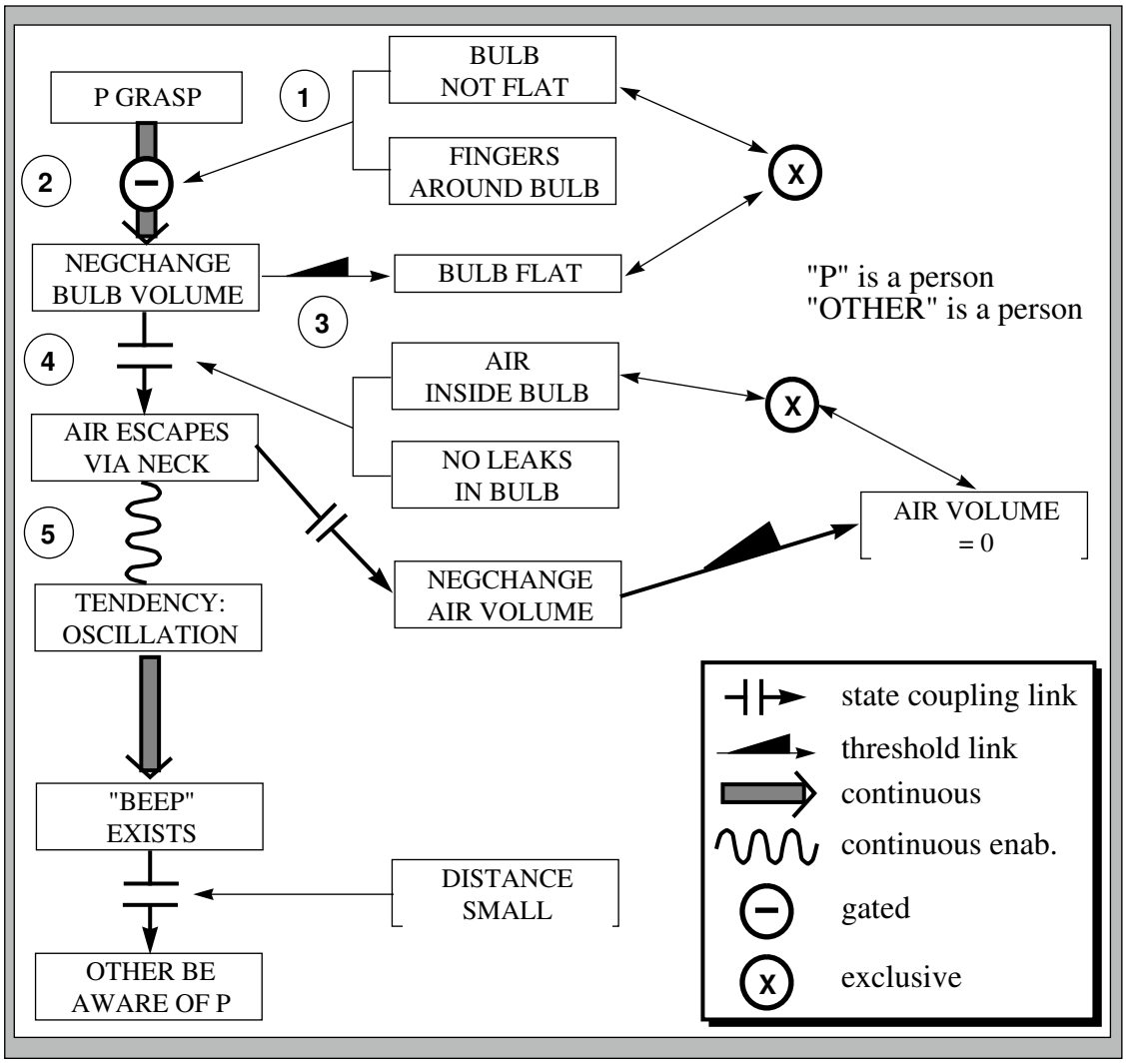


Figure 8.8 Rieger's CSA applied to a bicycle horn.

Figure 8.8 shows how a bulb bicycle horn works using Rieger's CSA. The central aspect of the approach was to identify 5 event types (action, state, statechange, tendency, and want), and to introduce 26 links which described causality, enablement, concurrency, iteration, intention, gating, and thresholding among those event types. In the figure, actions such as "P GRASP" and "OTHER BE AWARE OF P" are events which define the initial and final states of the horn's function. The GRASP event represents the input applied to the horn which initiates the function. The AWARE event represents a perceptual effect produced by the horn, which terminates the function. The actual function of the horn, in this context, can be considered the production of sound, and that OTHERS become aware of the sound and P, is related to the goals of P. For the most part, the boxes in the diagram represent states of various components of the bicycle horn. For example, "NEGCHANGE BULB VOLUME" describes a new state of bulb volume. The tendency "OSCILLATION" also describes a change of state, even though it appears to be a dynamic structure. The dynamics in this diagram/representation are captured by the links. For example, in the presence of P grasping the bulb, fingers being around the bulb, and the bulb not being flat (at 1), a one-shot, continuous state change is effected (2), that of the negative change in bulb

volume. The one-shot aspect of the relation means the events are sequential and non-repeating. The circle in the link is called a gate, and identifies additional enabling conditions for the event. The link at (3) represents the associated intended state that P has, and that, eventually, the negative change in bulb volume will lead to the bulb being flat. It is called a threshold link because the goal is being approached, and is suddenly achieved. The link at (4), called a state coupling link, relates two states through an unknown causal relation. It makes an equivalence between the negative change in bulb volume and the escaping of air through the neck of the horn. The relation at (5) is called a continuous enablement, and relates the escaping air to the oscillation tendency.

CSA emphasizes how objects interact, and how object functions are applied and perceived by problem solvers. It identifies the types of relationships which problem solvers associate with object use. The notion of causality is identified at the object function level and the attempt is to relate observed states with object behavior (tendencies). CSA and FONM can both be used to represent the function of a device and how function is applied and perceived by its user. CSA identifies object function with actions by the observable states which are produced by the function when the action is applied. FONM does this, but also identifies object function with the plans and goals associated with why the object function is being invoked. Thus a FONM representation supports the application of devices used in dissimilar situations as long as the goals or plans are similar. CSA's dynamic links provide a notion of temporal causality as interpreted by a device user, and has no direct equivalent in FONM. In FONM, temporal causality is captured by the sequencing of behavioral processes, which are enabled or disabled by the states of other objects. There is no FONM notion of a threshold, for example, because it implies the combination of causal effects (one thing happens, a value is reached, and then something else happens). In FONM each causal effect is associated with a particular process, so the thresholding value in CSA is an enabling condition in FONM. The same is true of gating conditions. The notions of one-shot and continuous enablement in CSA are related to FONM by how the behavioral process is defined. A FONM process, once enabled, remains enabled unless disabled by another state or process, so the same causal relationships are represented without defining new forms of causality.

8.2.4 Object Primitives and the Representation of Device Use

Lehnert's *Object Primitives* were proposed to assist in making inferences about objects in natural language processing, such as for the sentences shown below [Lehnert 1978]:

S1: John drank from the faucet.

Q1: What did John drink?
A1: Water.

S2: John filled his canteen at the spring.

Q2: What did John get at the spring?
A2: Water.

Lehnert's notion was that the faucet and spring both serve (from a naive perspective) as sources of water and should share representational similarities which enable a program to recognize this and make similar inferences. Lehnert proposed a set of 7 primitives for describing similarities and differences between objects based on how they are used: (1) setting, (2) gestalt, (3) relational, (4) source, (5) consumer, (6) connector, and (7) separator.

The use of these primitives are illustrated in the examples shown in Figs. 8.9 - 8.13.

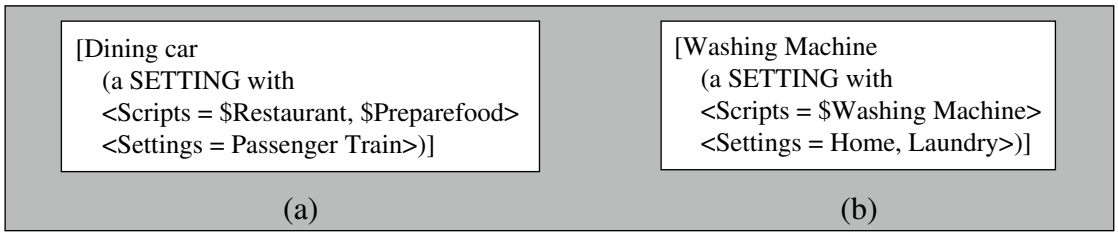


Figure 8.9 Examples of SETTINGS, (a) location type, (b) object type.

Fig. 8.9 illustrates the two types of SETTING in the Lehnert approach. In the first case, a setting is a location where something happens, and can be associated with scripts and additional settings. In the second case, a setting is identified by a particular object in terms the tasks which can be undertaken with it. This latter designation is similar to FONM REGIONs and machine primitives, which relate an object to its known functions, and with the plans which invoke a particular function. A device's intended function and its associated plan are then similar to the Lehnert functional setting.

Lehnert's notion of GESTALT is used to describe object compositions which are functionally related, whether or not they are connected together. Two examples are illustrated in Fig. 8.10

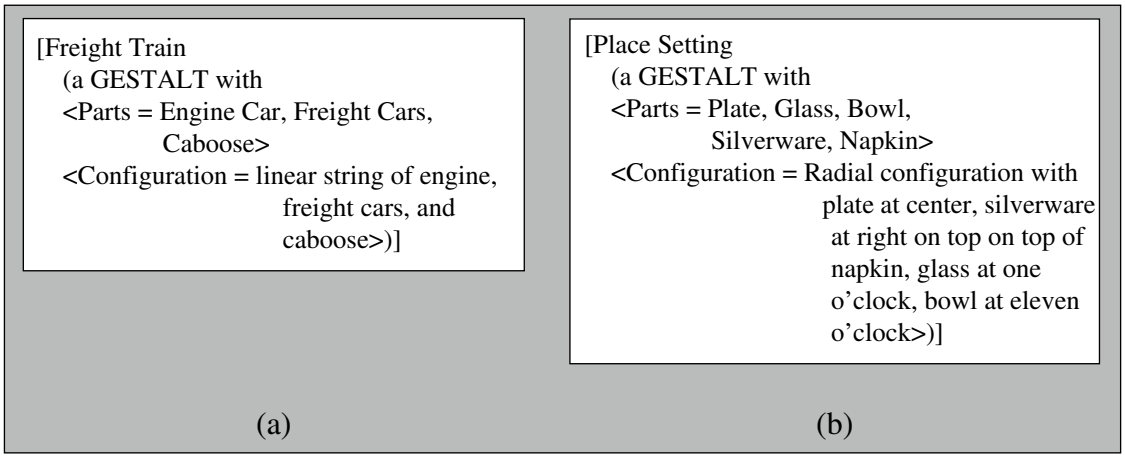


Figure 8.10 Examples of the GESTALT primitive, (a) connected objects, (b) functional cluster.

The GESTALT examples in Fig. 8.10 point out the need to identify an object's components and their orientation to one another. Unlike most representation approaches for objects, the Lehnert model has no specific representation for placement or connectivity. For example, in Fig. 8.10b, the manner in which orientation and value are defined is left unclear. In FONM, one object is selected as a focus, and the placements of the other objects are represented with respect to that object. This is an important consideration if component placement is important to how the device functions as a whole.

The RELATIONAL primitive describes an object's relationship with, and dependence upon, other objects, as shown in Fig. 8.11. In this figure, the RELATIONAL is used to represent object support (Fig. 8.11a) and object hanging (Fig. 8.11b). The representation provides the participants, and a link identifying their behavioral dependency (e.g., On-top-of and Stuck-to), but no representation for how these relationships are achieved. In FONM

support and hang are represented with RESTRAIN process instances. The relationlink 'On-

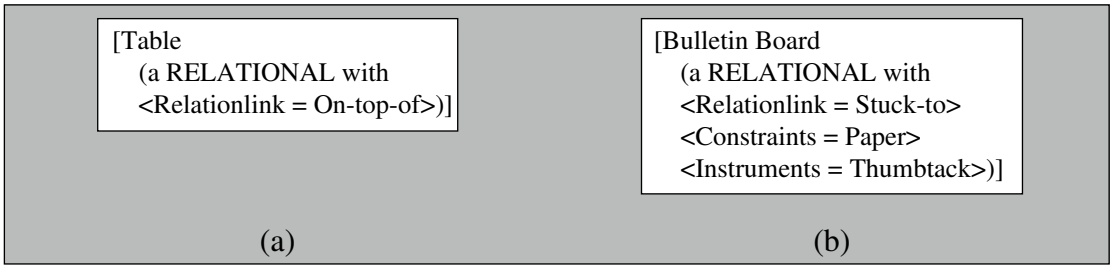


Figure 8.11 Examples of RELATIONAL, (a) table support, (b) bulletin-board and thumbtack.

top-of' is represented with an ORIENT structure which identifies the dimension of comparison and the value which defines the tables as below the object. A RESTRAIN-SUPPORT then represents the restraint state which defines the object's support dimension and location. Similarly, the 'Stuck-to' relationlink is represented, in FONM, with a RESTRAIN process which results in a restraint state on the paper in all dimensions accept rotational about the thumbtack axis. By not representing the dynamics associated with these relations, it is impossible to say anything about the behavior of the objects when the relation is removed. For example, if the thumbtack is removed in Fig. 8.11b, the RELATIONAL doesn't say whether the paper will fall. Also, the RELATIONAL doesn't show the similarities between the two relations (On-top-of and Stuck-to), which reduces the ability to share inferences between the two relations.

The SOURCE and CONSUMER primitives are used to infer context of object use by defining how the object is related to the context in which it is used. For example, a sponge can be represented in terms of providing a source of fluid when squeezed, and as a consumer of fluid when used to wipe (Fig. 8.12). Recognizing what action is being performed then enables an inference of which object function is being applied. In FONM a sponge would be represented as a container. The container can be filled or emptied, each of which has an analog with the notions of source and consumer.

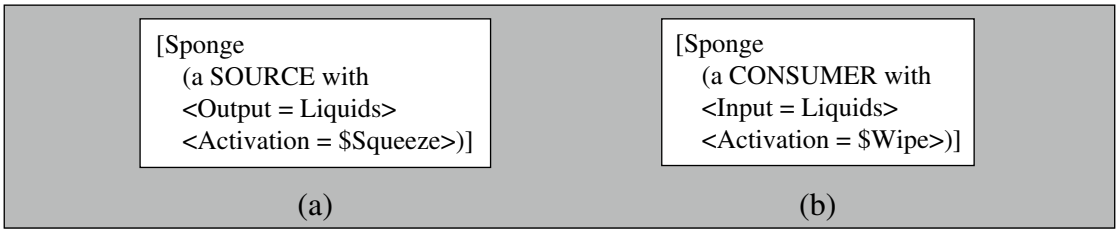


Figure 8.12 Example of a sponge as a SOURCE and CONSUMER

The SEPARATOR and CONNECTOR primitives enable and disable actions a user can engage in with respect to the object, and also assist in recognizing how the object is being used. A separator disables actions between spatial regions, and a connector enables actions between spatial regions. Consider the window represented in Fig. 8.13. When the window is open, communication (MTRANS), perceptual stimulus (SPEAK), perceptual reception

(ATTEND), and movement (PTRANS) are enabled. When the window is closed, both the

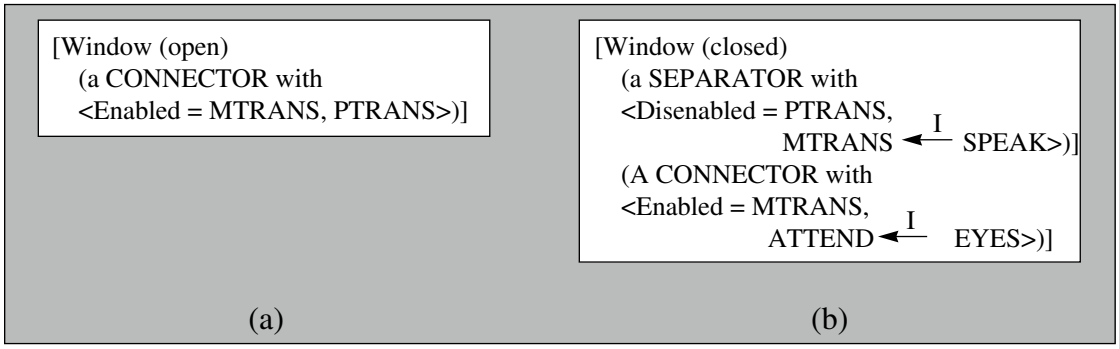


Figure 8.13 Examples of window as CONNECTOR and SEPARATOR, (a) open, (b) closed.

SPEAK and PTRANS actions are disabled. In FONM the window is represented as a container with opening and closing functions. The states which initiate and terminate the respective functions enable and disable PTRANS type actions, while the physical characteristics of the window enable and disable MTRANS and ATTEND type actions. The function is recognized by associating the goals of an actor with the existing states of the device. The difference between the approaches, for SOURCE/CONSUMER and CONNECTOR/SEPARATOR is that FONM associates functionality with the object and object primitives associate functionality with intent.

8.2.5 The Multilevel Flow Model and the Representation of Systems

The Multilevel Flow Model (MFM) has been proposed by Lind [Lind 1990, Lind et al, 1992] to model the function and control of complex dynamic systems. The MFM model is intended as an integration of system function and the intentions (i.e., goals) of the users who maintain and supervise its operation. The model describes a physical system as being comprised of: (1) goals, (2) functions, and (3) physical components. The overall system is hierarchically defined to describe subsystems and their causal integration. The model provides formalized concepts that describe causal mappings between these system components. In MFM, three types of goals are defined: (1) maintain, (2) prevent, and (3) achieve. These goals can be applied to one of three categories: (1) safety goals, (2) production goals, and (3) economy goals. Goals are represented as a 2-tuple: an expression to be met, such as an inequality, and a priority for ordering. The use of these goals is applied to an example

of a central heating system (Fig. 8.14):

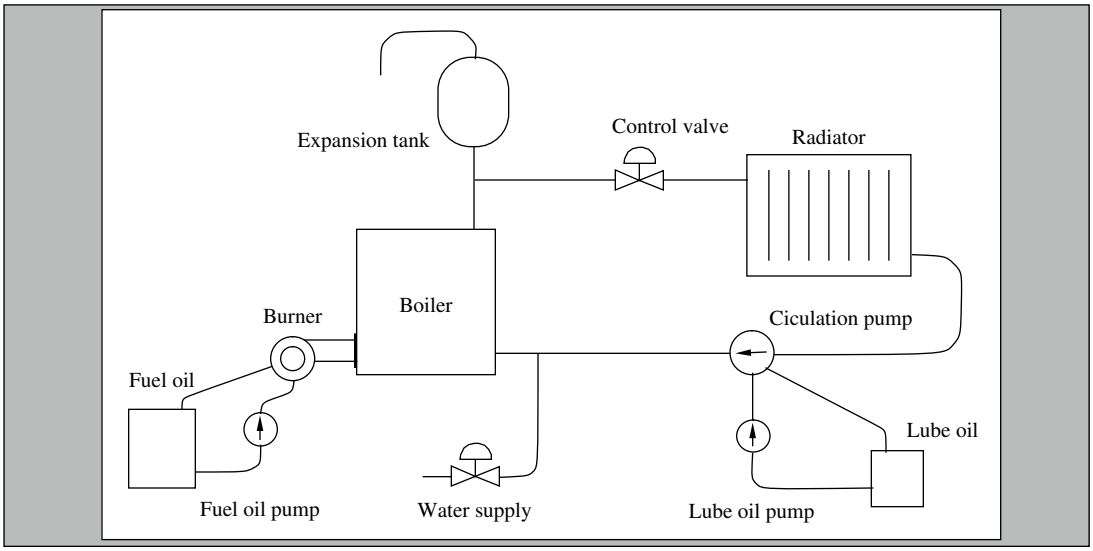


Figure 8.14 Central heating system example for Multilevel Flow Modeling.

Lind identifies 5 goals (G11, G21, G22, G31, G32) associated with the operation of this system:

G11: Maintain room temperature within defined limits

G21: Minimize heat losses

G22: Optimize fuel combustion

G31: Keep water temperature below boiling point

G32: Keep water inventory below upper limits

These goals can be categorized as production goals (G11), economy goals (G21 and G22), and safety goals (G31 and G32). The approach then defines the functions associated with the goals of the system. MFM has eleven functions associated with the achievement of these goals in the central heating system:

F1: Energy supply

F2: Storage of fuel

F3: Air-gas path in the boiler

F4: Distribution of heat in the boiler

F5: Circulation of water

F6: Transfer of heat from boiler to radiator

F7: Control of room temperature

F8: Lubrication of the circulation pump

F9: Storage of lubrication oil

F10: Circulation of lubrication oil

F11: Water supply

In MFM, function is also represented as relations to be met. Three types of function are

described: (1) mass and energy functions, (2) information functions, and (3) organizational functions. The mass and energy functions describe the functional relationships between components and systems, while the other two function types describe how the information associated with device and system function is propagated or controlled. Each of the functions in F1-F11 represent mass and energy functions associated with fluid flow in the central heating system model.

At the lowest level are the physical components themselves. There are fourteen components in the central heating system, most of which are shown in Fig. 8.14.

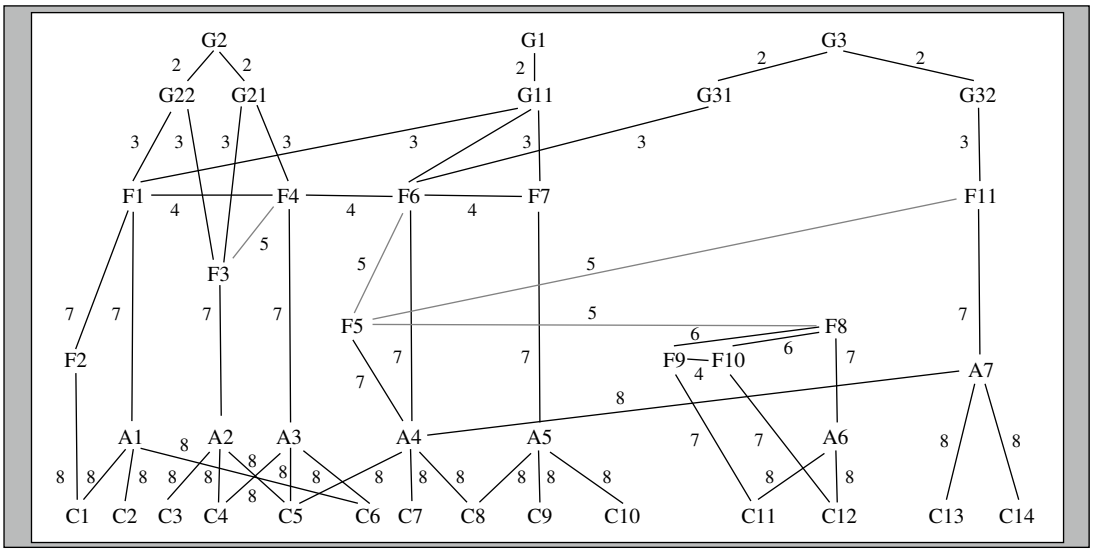


Figure 8.15 MFM mappings between goal, function, and component levels for a central heating system.

The model describes causal mappings between each of these component levels, as shown in Fig. 8.15. The items `G1-G32` represent goals, `F1-F11` represent functions, `A1-A7` represent component aggregates (like mechanisms), and `C1-C14` represent the low-level components. The arcs are labelled with numbers which describe four types of relationships between component levels: (1) ordering, (2) decomposition, (3) relations, and (4) connectivity between the components as detailed below.

1) **Ordering** of goals according to priority
2) **Decomposition** of goal into subgoals
3) **Relation** between a goal and the functions which can achieve the goal
4) **Connection** of two functions belonging to the same level
5) **Relation** between a function and its supporting function
6) **Decomposition** of function into subfunctions
7) **Relation** between a function and its physical implementation
8) **Decomposition** of an aggregate into subaggregates or components
9) **Connection** between two components

In MFM, these mappings are expressed graphically. For example, the graphical repre-

sentations for goals and functions for mass and energy are shown in Fig. 8.16:

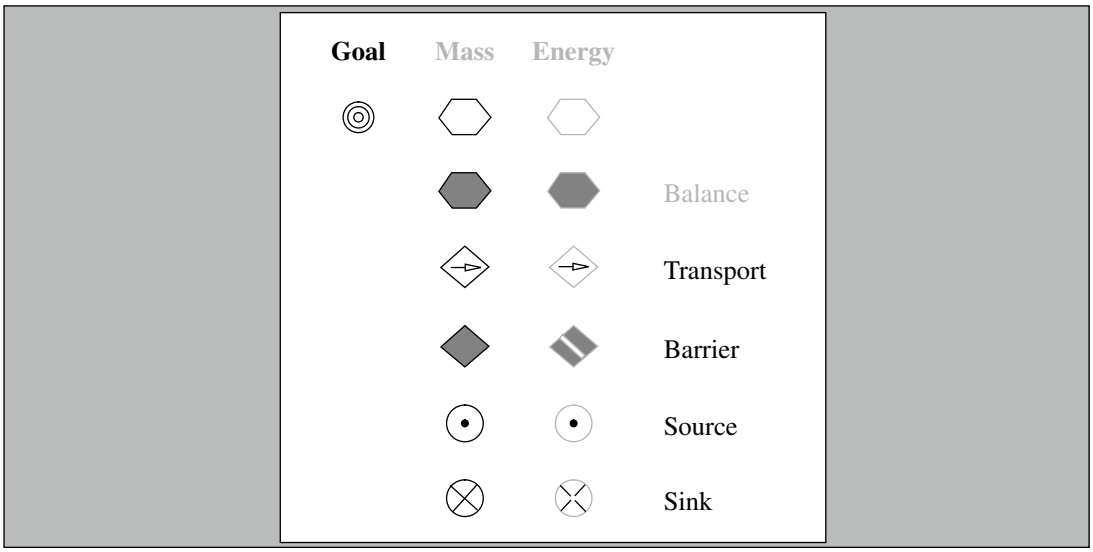


Figure 8.16 Graphical representations of MFM mass and energy flow functions.

These functions represent the functional building blocks for constructing MFM flow models. The functions are related to goals (and components) with links. An 'A' link represents the achievement of a goal with a function, and an 'A-C' link represents the achievement of a goal which has a controlling influence on a function (e.g., the addition of a controller unit). There are rules which define which functions can be combined and how they can be combined to maintain physical consistency.

The MFM model of the central heating system is shown in Fig. 8.17. The major func-

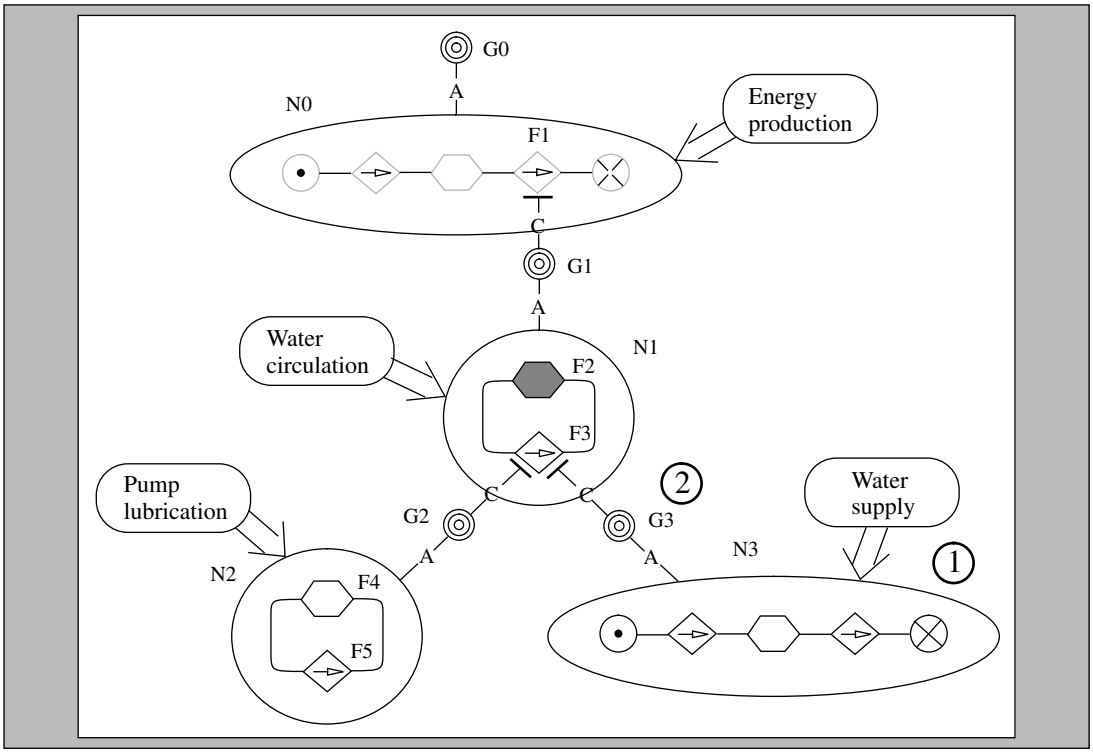


Figure 8.17 MFM model for the central heating system.

tional components can be seen to comprise their own flow subunits (`N0–N3`). The functions of these aggregates achieve subgoals of the overall system's control. For example, the `water supply` function (at 1) achieves the goal `G3` (2), which itself is a condition for the `water circulation` (transport) function. Likewise, the `pump lubrication` function achieves the goal `G2`, which itself is also a condition for water transport (i.e., the pump must be lubricated in order to function). The function `water circulation` then achieves the goal `G1` to provide water flow in the system, without which temperature cannot be controlled. `G1` is a condition on the function of `energy production`, which achieves the overall goal `G0`.

MFM and FONM are very similar in design philosophy. Both models approach the representation of function from the standpoint of how the device is used by individuals and by how the device behaves in lieu of individuals when perturbed. FONM uses device-use plans to provide an intermediate step between user goals and the functions which, when invoked, aid in their achievement. Device-use goals satisfy the control and organization functions in MFM which, in both models, can be achieved by a human or a device. The MFM model does not represent the actual behavior of devices, but, rather, the relations which are met when the device is functioning properly. As such, the diagnostic capabilities of the model are limited. The notion of a flow module is very similar to that of a machine primitive. That the functions represented in MFM are based on mass and energy, and, in particular, with transport and storage, suggests that FONM and MFM models could be merged to describe mechanical and fluid mechanical devices.

8.2.6 Bondgraphs and the Construction of Aggregate Behavioral Models

A *bondgraph* is a graph model for describing energy exchange between components and was developed as an approach for constructing device models for conceptual design [Ulrich and Seering, 1987]. A bondgraph is comprised of ports and bonds, where a port is an object type and a bond represents a path of energy flow between ports. Each bond has associated with it two variables: (1) effort, and (2) flow. In mechanical systems effort represents force, and flow represents velocity. There are four port types in this model: sources, 1-ports, 2-ports, and n-ports. A source represents a device which specifies a force or velocity. A 1-port describes devices which, when a force is applied, have a behavior which is based on force. A lever and a spring would be considered 1-ports, because the type of input and output are basically the same. A 2-port describes objects which transform or convert their input. A screw is a 2-port, because it converts rotational motion into translational motion. N-ports represent bond junctions. There are two types of n-ports: one junctions and zero junctions. A one junction represents a junction of bonds which share a common energy flow, while a zero junction represents a junction of bonds which share a common effort. The sum of the efforts at a one junction is zero, while the sum of flows at a zero junction is zero.

Bondgraphs are constructed by connecting ports and bonds, and can be combined by connecting bondgraph chunks between n-ports. There are four ways to construct a bondgraph:

(1) A source connected to an n-port

(2) A 1-port connected to an n-port

(3) A 2-port connected to an n-port at each port

(4) Two bondgraphs connected by bonds between n-ports

A partial table illustrating some of the bondgraph elements proposed by Ulrich and

Seering is shown in Fig. 8.18:

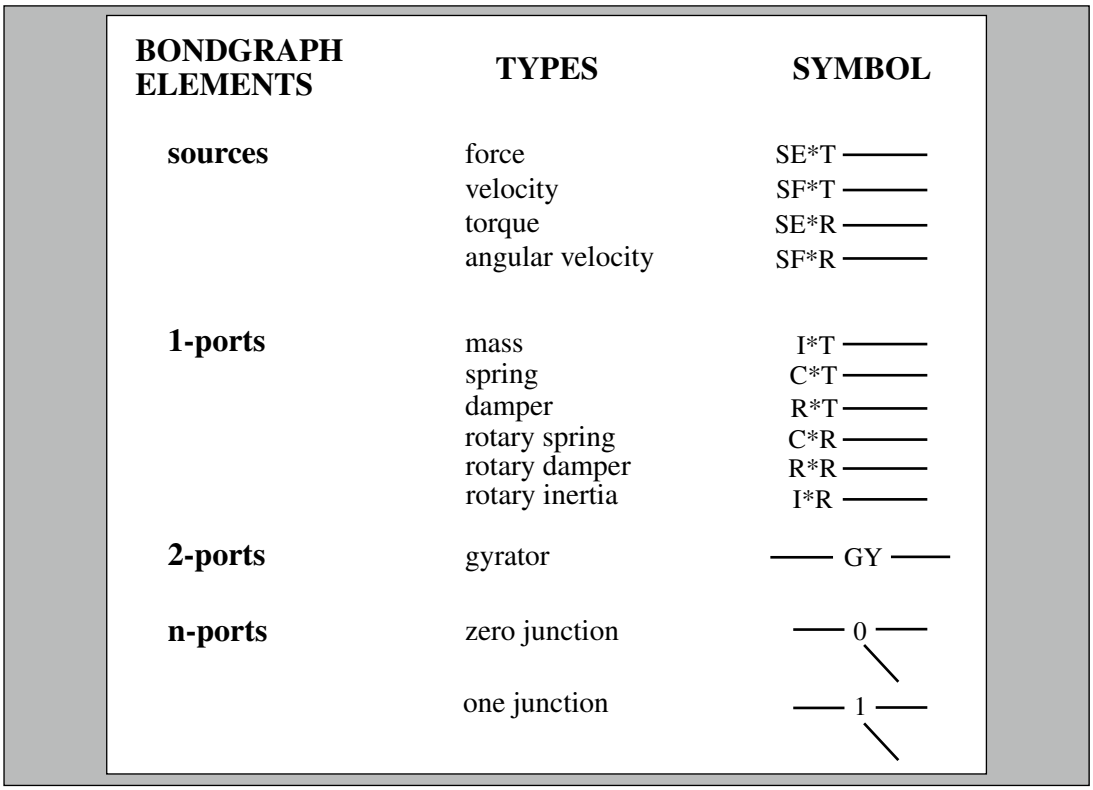


Figure 8.18 Bondgraph elements for simply mechanical objects.

A simple illustration of the use of bondgraphs for representing a mass, spring, damper

system is shown in Fig. 8.19. The spring is moving with a velocity SF*T (at 1), which is

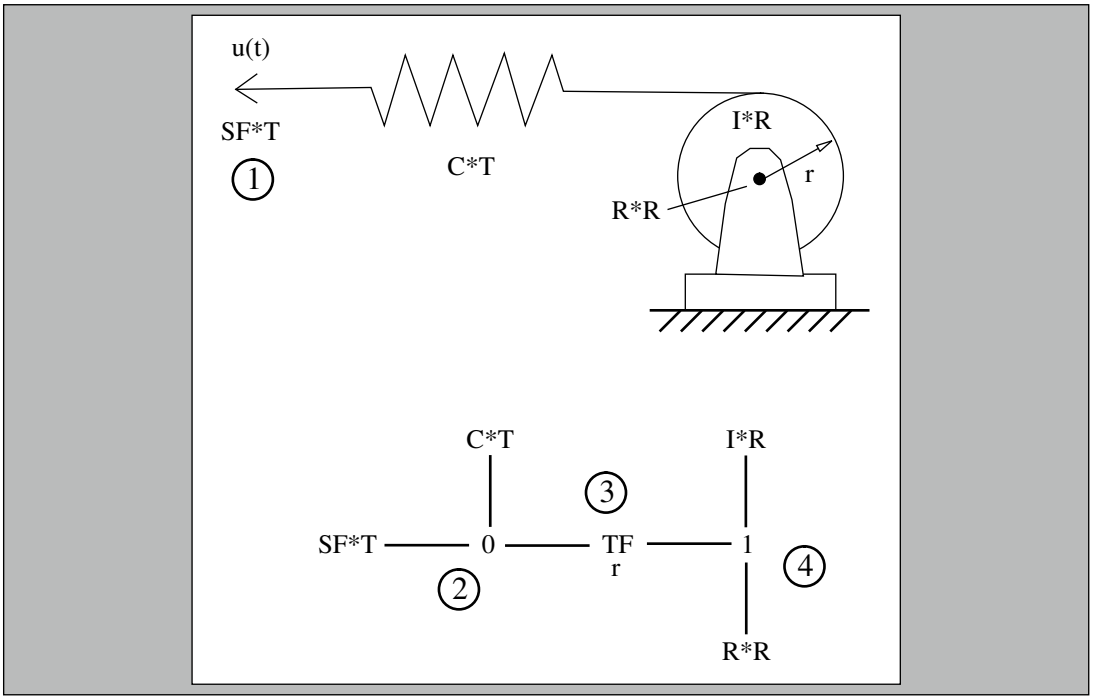


Figure 8.19 Bondgraph representation of mass, spring, damper system.

represented as a source. The spring constant of the spring is represented as a 1-port describing the storage of internal energy, `C*T`. The source and spring 1-port are connected to a zero junction (at 2), as well as to a transformer (2-port, at 3). The zero junction describes the conservation of flow. The transformer is also connected to a one junction (at 4), which is connected to two 1-ports: one representing rotational inertia (`I*R`) of the wheel, and the other representing the rotational damping (`R*R`) of the wheel.

The bondgraph approach is a behavioral analog to FONM machine primitives. The bondgraph can be used to construct an overall behavioral description of the device, by constructing device models using the rules for combining bondgraph segments and knowing something about the desired input/output behavior. The machine primitive decomposes to a behavior representation of the related device function; however, the primitive is intended to be used in constructing devices where the behavioral description is not deemed as important as what the device can do and what goals it can be applied to, so the behavioral aspect of the FONM representation is used for explanation rather than generation. Although the bondgraph approach is appealing in its ability to describe the dynamic relationships between components, it has no model of use embedded in it, so the devices which are constructed can only be compared using structural and behavioral characteristics (i.e., MP-Equivalence).

8.2.7 The Finite Element Method

The Finite Element Method (FEM) is an approach for quantitatively representing complex nonlinear systems as a combination of discrete linear constructs [Clough 1960], where the causal mapping between constructs mandates compliance with engineering mechanics. FONM is similar to FEM in four ways: (1) design of discrete individuals, (2) type of ele-

Appendix
Mechanical Device Representations

This appendix is devoted to an in-depth presentation of the static, dynamic, and pragmatic representations for each of the devices that have been used in this research. Fig. A.1 illustrates the four mechanical device classes which are presented in this appendix: (1) simple devices, (2) simple compound devices, (3) multiple compound devices, and (4) complex devices

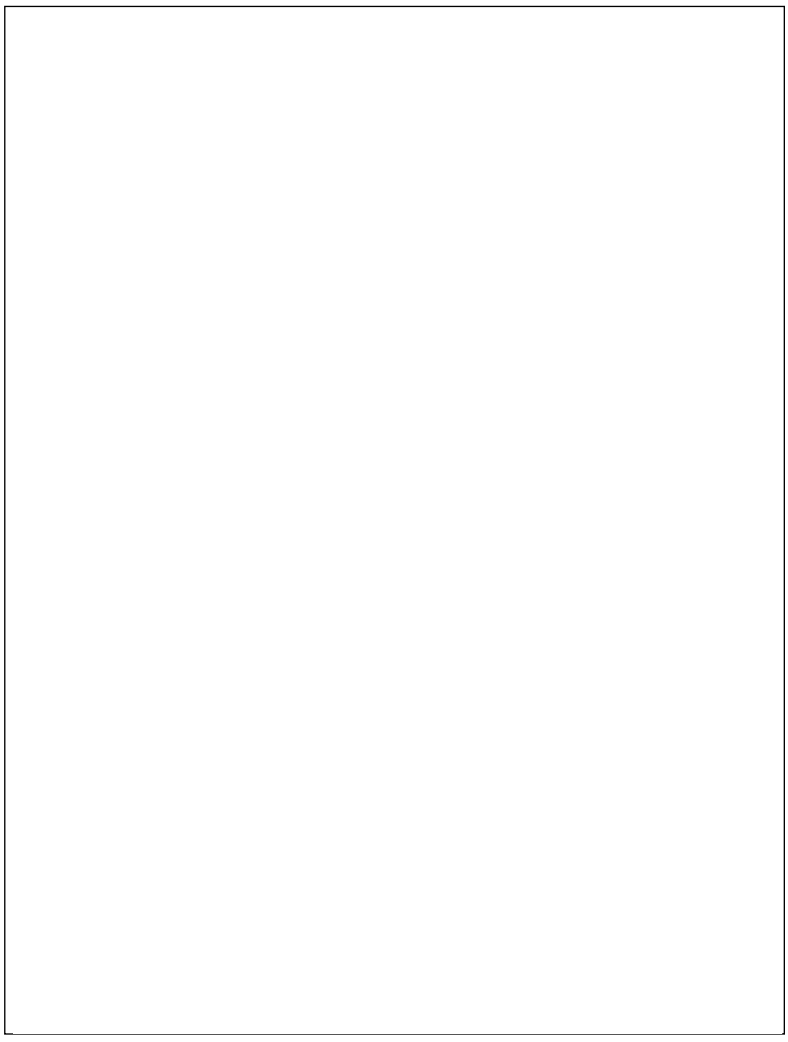


Figure A.1 FONM Devices.

A.1 Simple Devices

Simple device dynamics are representationally significant because even simple devices can be used to accomplish multiple tasks. The dynamic representation must make a device's different functions explicit with the same static representation. The distinction between device functions is often seen as a specialization on the type of input applied to a device, or