PART III

Related Work, Scope, Extensions and Conclusions

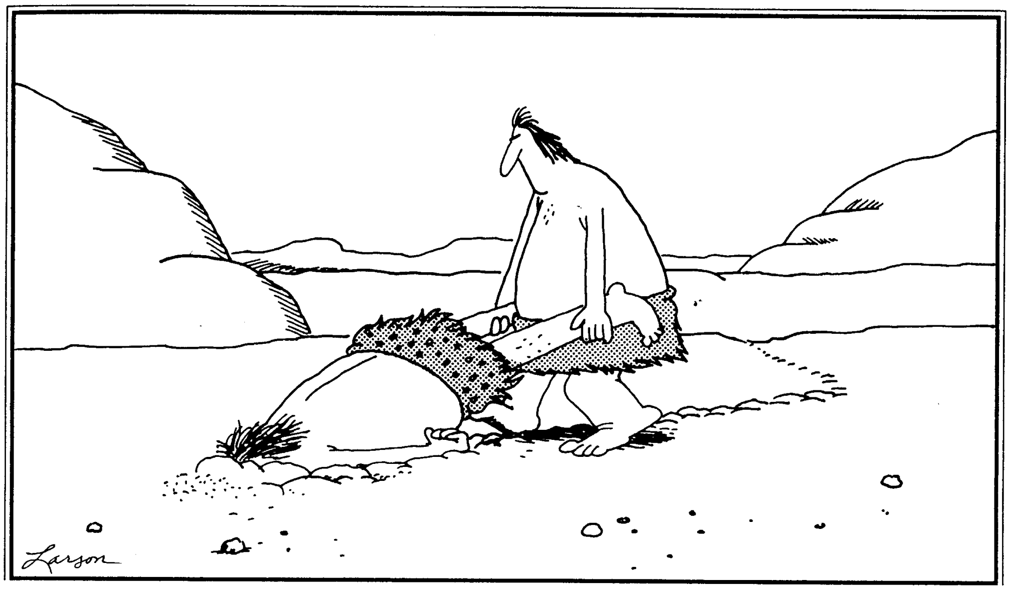
The value associated with the FONM representation theory is determined as a function of how it relates to existing work, how well it addresses the problems it is intended to address, and to what extent it provides new knowledge in a rapidly evolving discipline. This part of the dissertation consists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

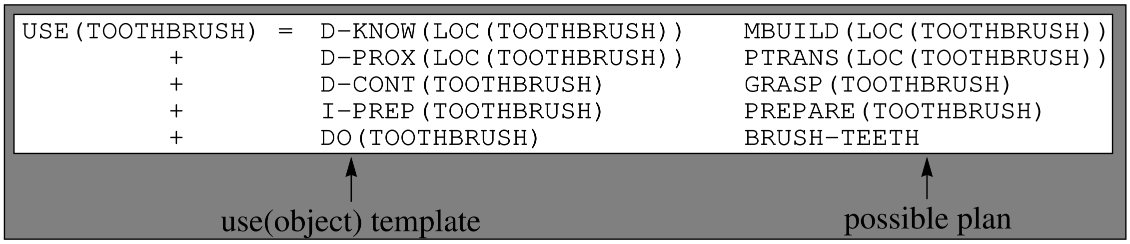
$$\mathrm{USE(object)} = \mathrm{FIND(object)} + \mathrm{D\text{-}CONT(object)} + \mathrm{I\text{-}PREP(object)} + \mathrm{DO} \tag{8-1}$$

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:

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

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



```
USE(TOOTHBRUSH) =  D-KNOW(LOC(TOOTHBRUSH))     MBUILD(LOC(TOOTHBRUSH))
          +        D-PROX(LOC(TOOTHBRUSH))     PTRANS(LOC(TOOTHBRUSH))
          +        D-CONT(TOOTHBRUSH)          GRASP(TOOTHBRUSH)
          +        I-PREP(TOOTHBRUSH)          PREPARE(TOOTHBRUSH)
          +        DO(TOOTHBRUSH)              BRUSH-TEETH
```

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 [Forbus 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

Object(P)

has-quantity(P, amount-of) and
has-quantity(P, volume) and
has-quantity(P, pressure) and
has-quantity(P, temperature) and
has-quantity(P, heat) and
has-quantity(P, t-boil) and
has-quantity(P, t-melt) and
substance(madeof(P)) and
place(location(P)), amount-of) and
temperature(P) and
heat(P)

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.

Individual View Moving-Friction

individuals:
    B an object, Mobile(B)
    S a surface
    dir a direction

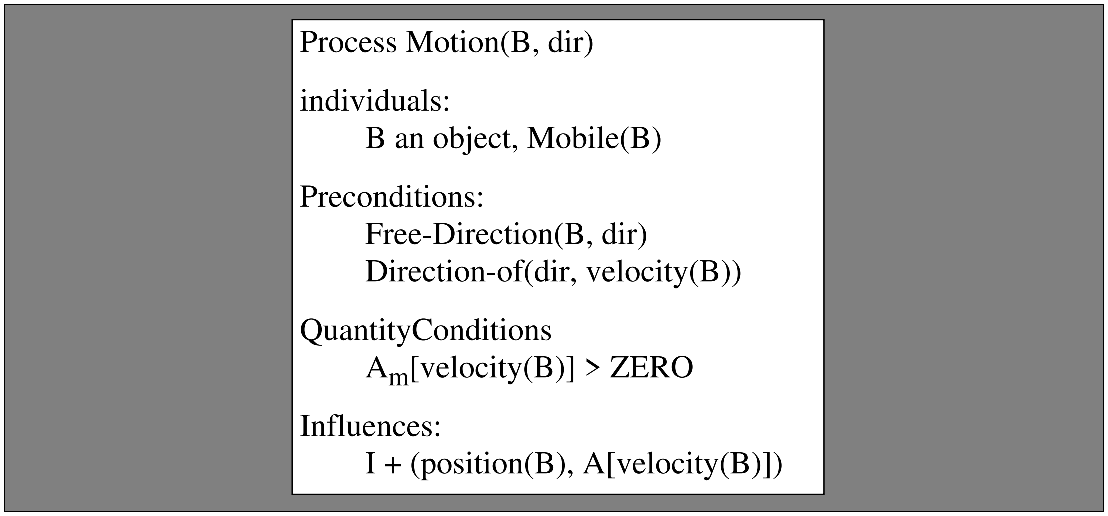
Preconditions:
    Sliding-Contact(B, S)

QuantityConditions
    Motion(B, dir)

Relations:
    Let fr be a quantity
    fr $\propto$+ normal-force(B)
    $A_S[fr] = -A_S[velocity(B)]$
    fr $\in$ forces-on(B)

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.



```
Process Motion(B, dir)

individuals:
    B an object, Mobile(B)

Preconditions:
    Free-Direction(B, dir)
    Direction-of(dir, velocity(B))

QuantityConditions
    $A_m$[velocity(B)] > ZERO

Influences:
    I + (position(B), A[velocity(B)])
```

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

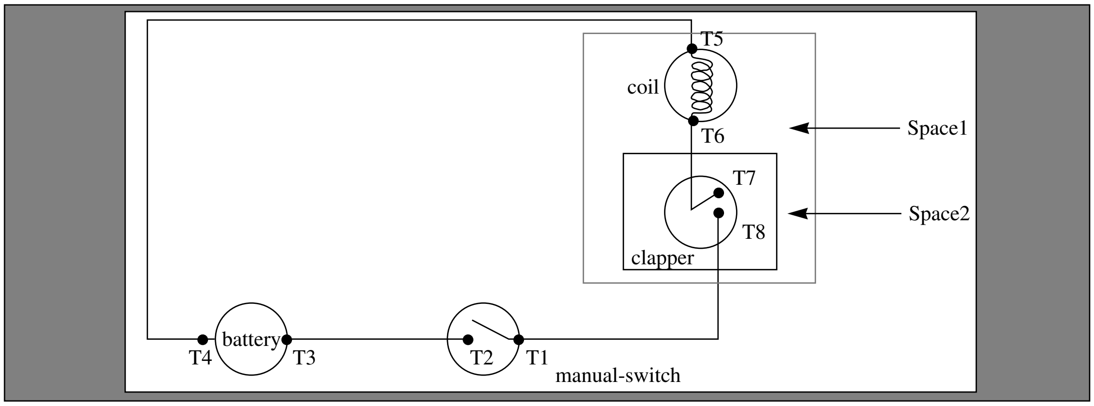


Figure 8.5 A schematic diagram of a household buzzer.

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

```
DEVICE buzzer
  FUNCTION
    buzz: TOMAKE buzzing (buzzer)
          IF pressed (manual-switch)*                              ①
          PROVIDED INITIAL elect-connected (T7, T8)                ②
          BY behavior1
    stop-buzz: TOMAKE ⌉buzzing (buzzer)                            ③
               IF ⌉pressed (manual-switch)
               PROVIDED INITIAL buzzing (buzzer)
               BY behavior5

  STRUCTURE:
    COMPONENTS:
      manual-switch (T1, T2), battery (T3, T4),
      coil (T5, T6, Space1), clapper (T7, T8, Space2)

    RELATIONS:
      serially-connected (manual-switch, battery, coil, clapper),  ④
      includes(Space1, Space2)

    ABSTRACTIONS-OF-COMPONENTS:
      COMPONENT clapper (T1, T2, Space)
        FUNCTIONS: mechanical, acoustic, magnetic                  ⑤
        STATES: elect-connected (T1, T2), repeated-hit (clapper)
      COMPONENT coil (T1, T2, Space)
        FUNCTIONS: magnetic
        STATES: magnetized (SPACE), voltage-applied (T1, T2)
      COMPONENT manual-switch (T1, T2)
        FUNCTIONS: connect
        STATES: elect-connected (T1, T2), pressed (manual-switch)
      COMPONENT battery (T1, T2)
        FUNCTIONS: voltage
```

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 components of the buzzer representation are shown in Fig. 8.7

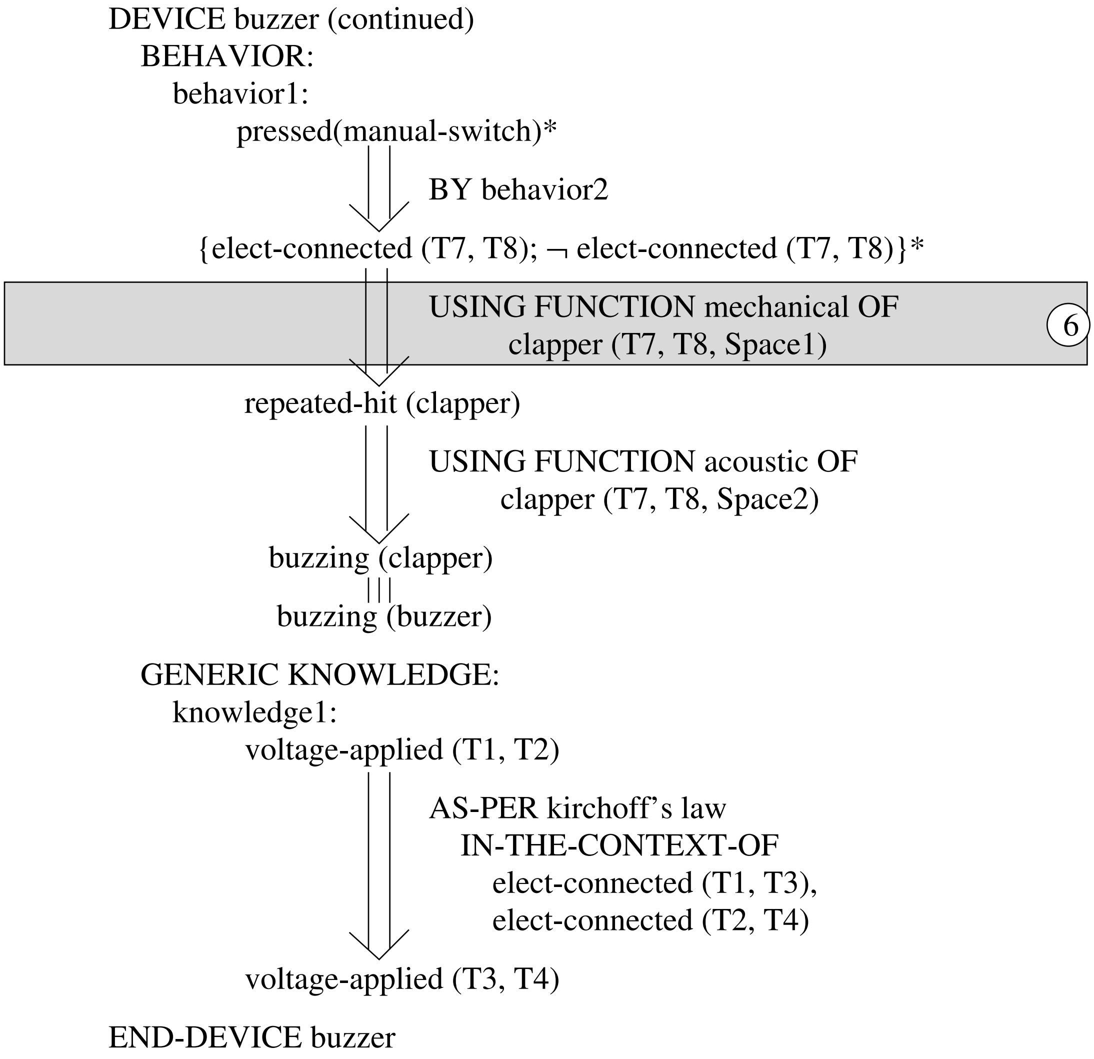


Figure 8.7 FR behavior and generic knowledge representation components of the electric buzzer.

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

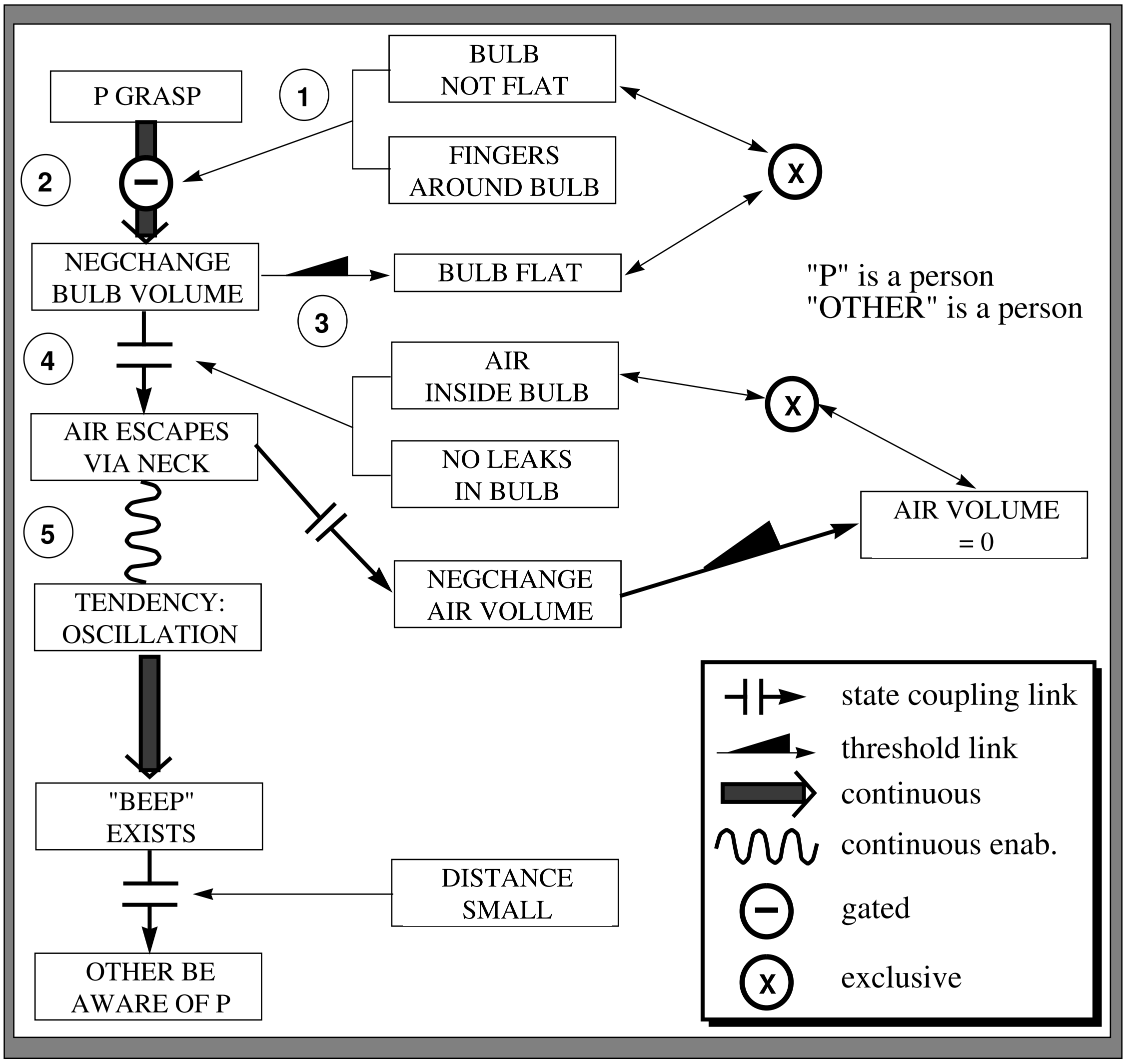


Figure 8.8 Rieger’s CSA applied to a bicycle horn.

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.

[Dining car
(a SETTING with
<Scripts = $Restaurant, $Preparefood>
<Settings = Passenger Train>)]

(a)

[Washing Machine
(a SETTING with
<Scripts = $Washing Machine>
<Settings = Home, Laundry>)]

(b)

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

[Freight Train
(a GESTALT with
<Parts = Engine Car, Freight Cars,
Caboose>
<Configuration = linear string of engine,
freight cars, and
caboose>)]

(a)

[Place Setting
(a GESTALT with
<Parts = Plate, Glass, Bowl,
Silverware, Napkin>
<Configuration = Radial configuration with
plate at center, silverware
at right on top on top of
napkin, glass at one
o'clock, bowl at eleven
o'clock>)]

(b)

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-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.

```
[Table
   (a RELATIONAL with
   <Relationlink = On-top-of>)]
```

(a)

```
[Bulletin Board
   (a RELATIONAL with
   <Relationlink = Stuck-to>
   <Constraints = Paper>
   <Instruments = Thumbtack>)]
```

(b)

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

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.

```
[Sponge
   (a SOURCE with
   <Output = Liquids>
   <Activation = $Squeeze>)]
```

(a)

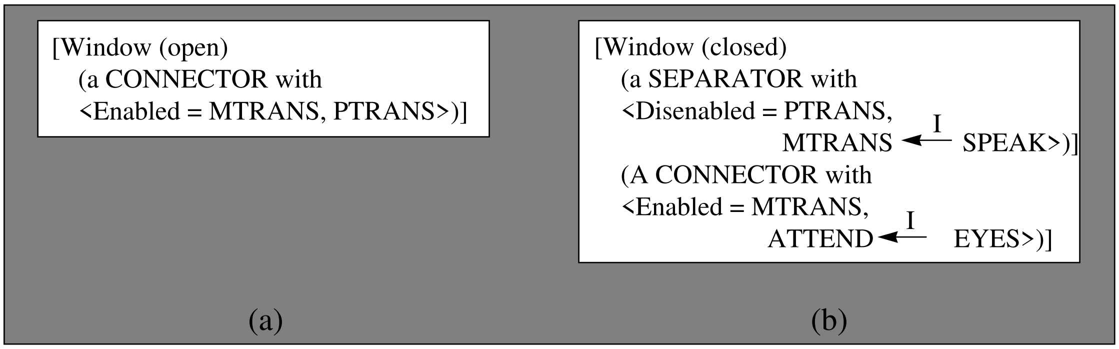
```
[Sponge
   (a CONSUMER with
   <Input = Liquids>
   <Activation = $Wipe>)]
```

(b)

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 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.



[Window (open)
(a CONNECTOR with
<Enabled = MTRANS, PTRANS>)]

(a)

[Window (closed)
(a SEPARATOR with
<Disenabled = PTRANS,
MTRANS $\xleftarrow{I}$ SPEAK>)]
(A CONNECTOR with
<Enabled = MTRANS,
ATTEND $\xleftarrow{I}$ EYES>)]

(b)

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

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):

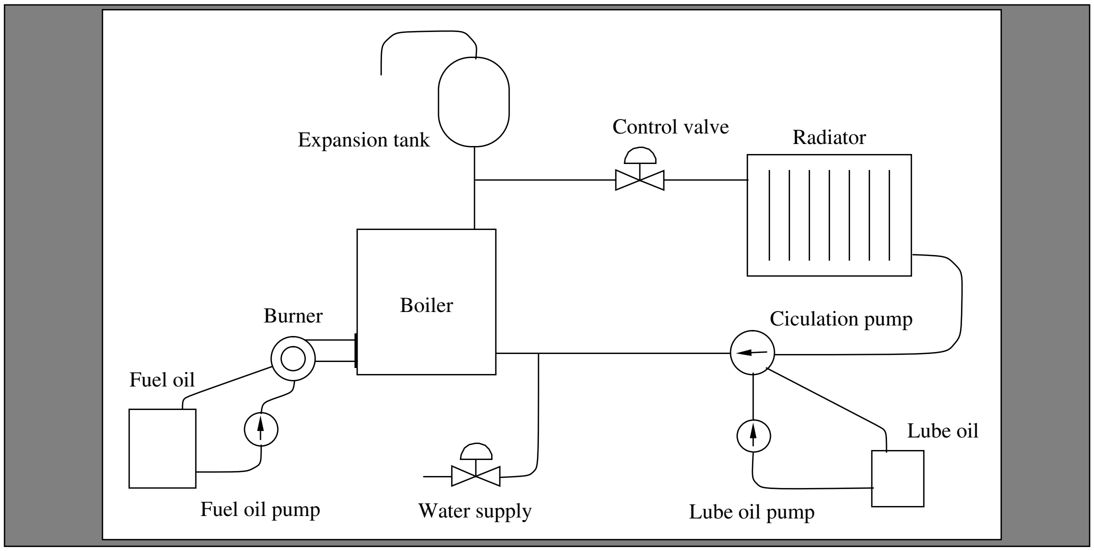


Figure 8.14 Central heating system example for Multilevel Flow Modeling.

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.

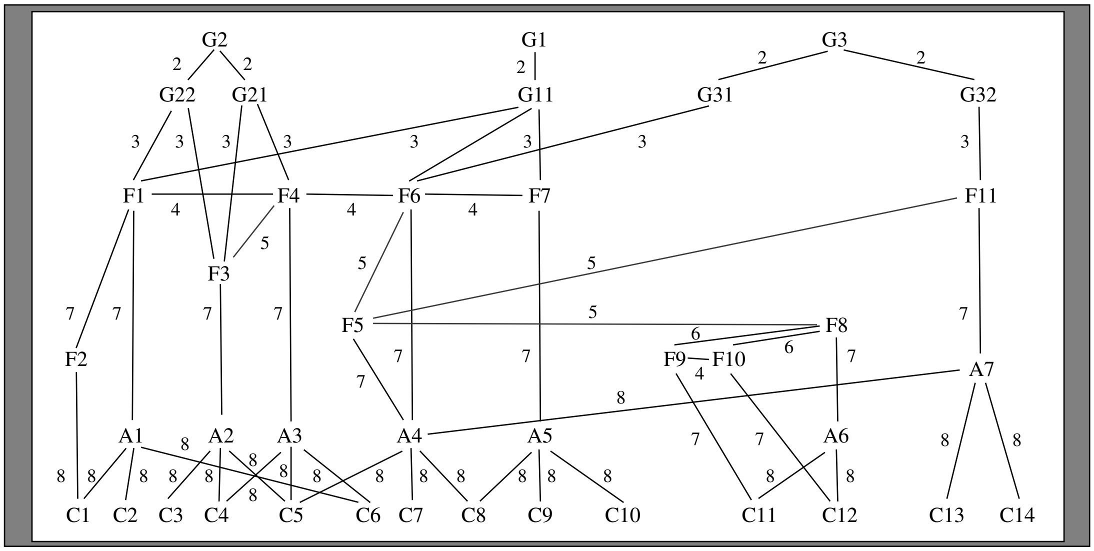


Figure 8.15 MFM mappings between goal, function, and component levels for a central heating system.

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

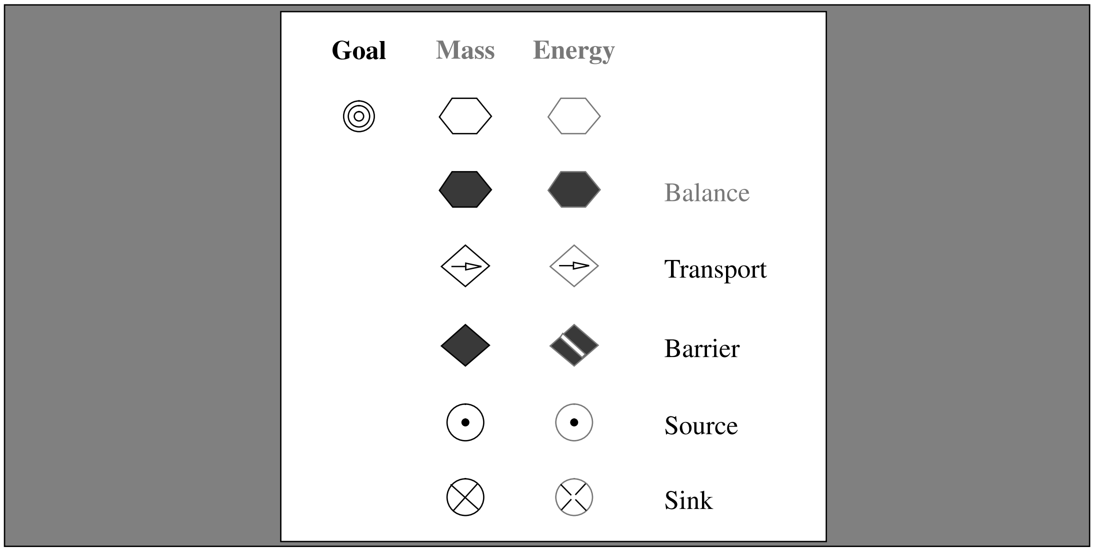


Figure 8.16 Graphical representations of MFM mass and energy flow functions.

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 functional 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`.

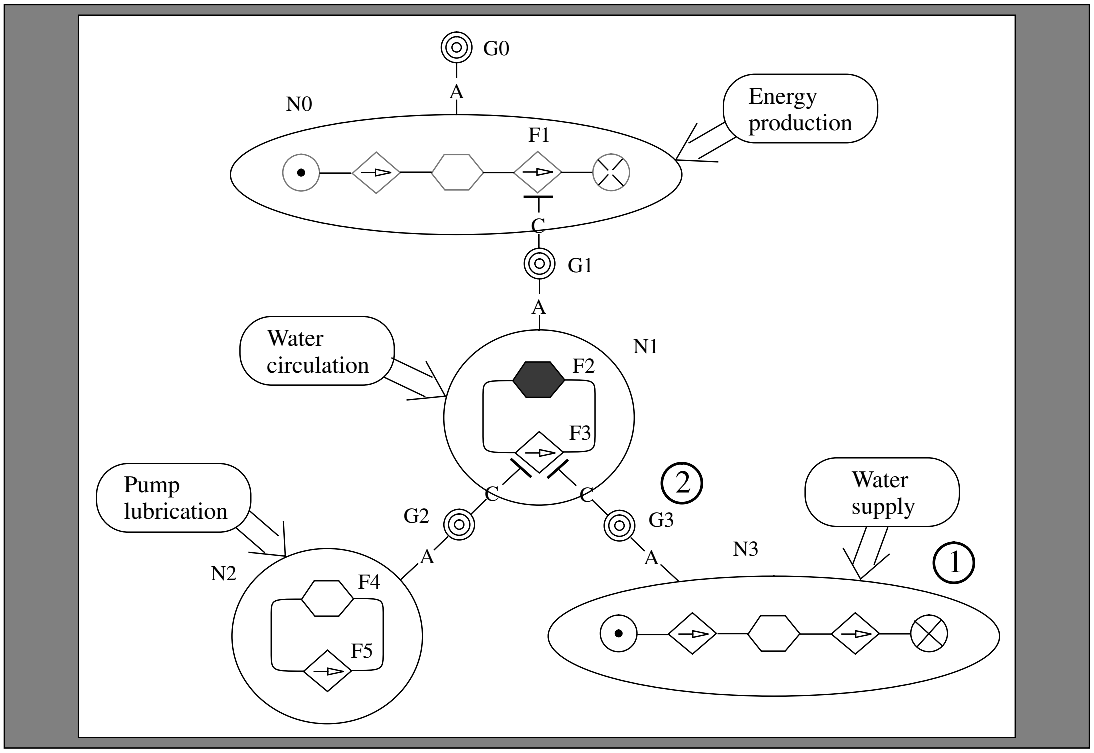


Figure 8.17 MFM model for the central heating system.

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

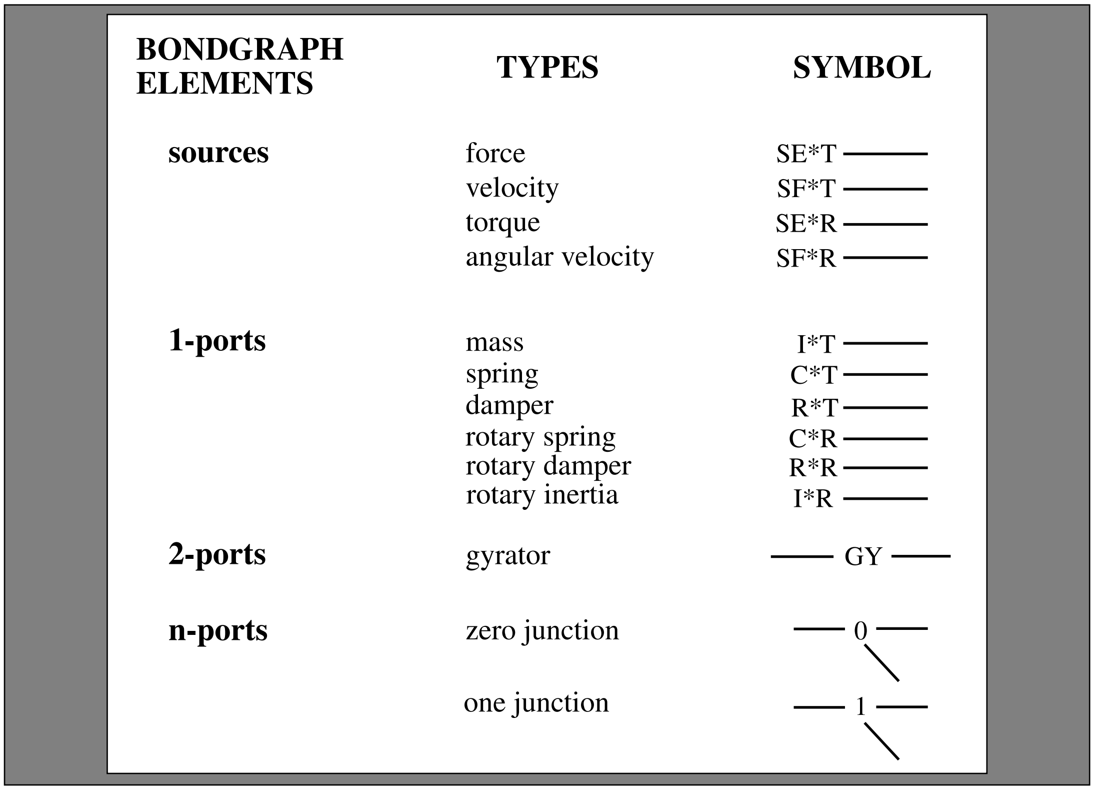
(2) A 1-port connected to an n-port

(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:



BONDGRAPH ELEMENTS	TYPES	SYMBOL
sources	force	SE*T —
	velocity	SF*T —
	torque	SE*R —
	angular velocity	SF*R —
1-ports	mass	I*T —
	spring	C*T —
	damper	R*T —
	rotary spring	C*R —
	rotary damper	R*R —
	rotary inertia	I*R —
2-ports	gyrator	— GY —
n-ports	zero junction	— 0 —
	one junction	— 1 —

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

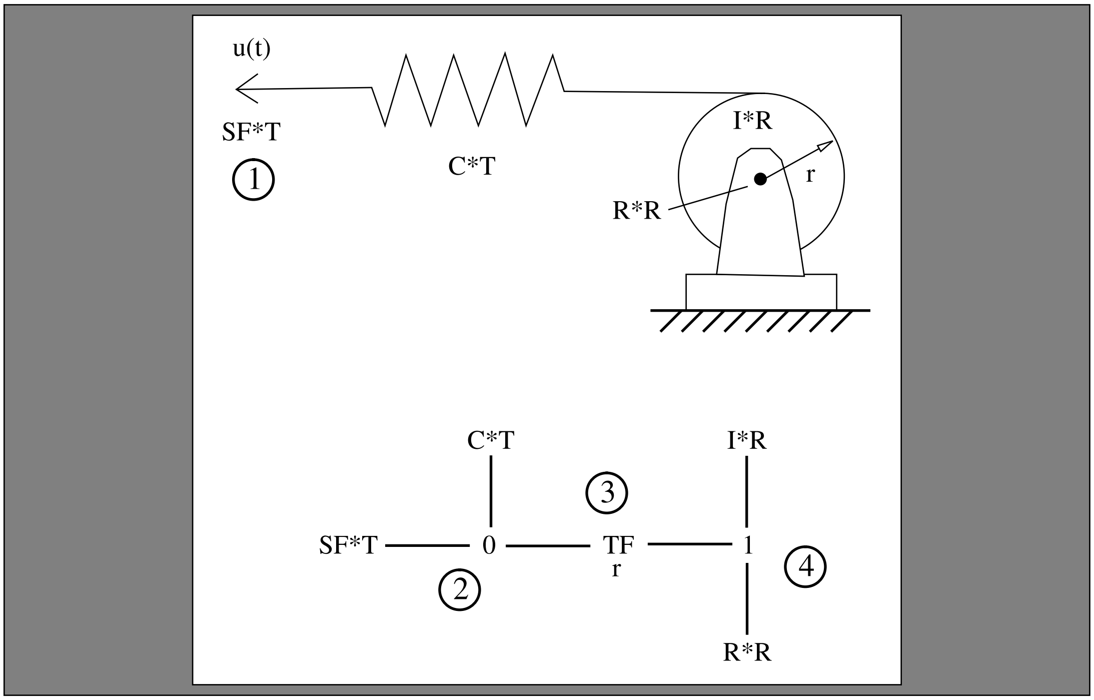


Figure 8.19 Bondgraph representation of mass, spring, damper system.

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 element reference, (3) coordinate systems, and (4) definition of behavior, and behavioral consistency/continuity.

Discrete Individual Objects

In FEM, a one dimensional element (such as a beam) may be represented as an idealized column (illustrated as a line) whose ends are called *nodal points*, as shown in Fig. 8.20.

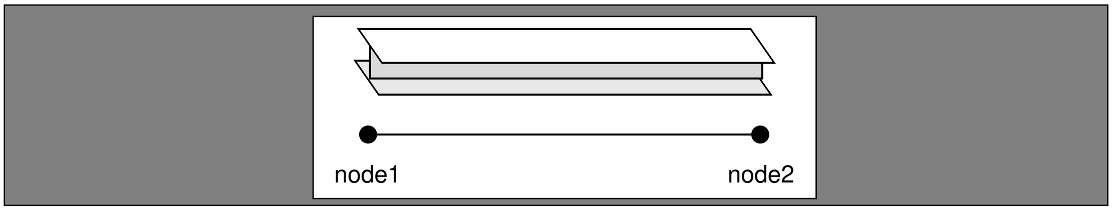


Figure 8.20 A single element and external nodes 1 and 2.

These nodal points, labeled node1 and node2 in the figure, are called *external nodes* because they represent connecting points to adjacent elements [Desai and Abel, 1972]. In the finite element method, behavior of an object is predicted as the combined behavior of the set of discrete elements, and the behavior of each element is effected only at its external nodal locations. FONM object primitives are similar to FEM elements in that (apart from the object CG) only external/observable locations are represented. In this respect, the FONM object primitive can be likened to a qualitative finite element although the granularity is much more coarse.

Nodal Reference System

The nodal displacements, rotations, and/or strains necessary to specify, completely, the deformation of a finite element are defined at the element *degrees of freedom* [Desai and Abel, 1972, page 84]. Element degrees of freedom differ from generalized coordinates in that each is specifically identified with a single nodal point and represents a displacement, rotation, or strain. In the FEM, degrees of freedom at external nodes are referred to as *joint* or *nodal degrees of freedom.* FONM degrees of freedom and FEM nodal degrees of freedom are equivalent.

Local and Global Coordinate Systems

A *local coordinate system* is one that is defined for a particular element and not necessarily for the entire body or structure; the coordinate system for the entire body is called the *global system* [Desai and Abel, 1972, page 88]. The FONM approach is based on a local coordinate system for objects; however, the notion of local is associated with each region (nodal point) rather than with an element, since in FONM an element and an object are one and the same. Object properties are described for an entire individual object. Forces and restraints are local (nodal) effects and are described locally, at object regions, and globally with respect to the entire object.

In FEM, the summation of nodal forces and moments at an arbitrary point (all degrees of freedom) is zero [Archer et al, 1972]. In FONM, the summation of nodal forces is represented as a union of local forces. The forces are resolved into the six Cartesian dimension equivalents used in FONM, and compared to local restraints to determine whether the object can move.

Object Behavior, Consistency and Compatibility

The FONM representation of object behavior is restraint-based. Object displacements and continuity relations can be described by the consolidation of nodal degrees of freedom. Degrees of freedom which are restrained cannot move, while those which are unrestrained are free to move.

FONM and FEM both describe objects as compositions of discrete elements. At some descriptive level, both approaches must agree with a continuous object model. The difference is in the type of model and the granularity of the analysis. *Granularity* refers both to the size of the elements and to the values of the resulting behavior. The finite element approach is a quantitative model, and is distinguished from continuous mechanical models by the discretization of objects into discrete linear entities. As the element size decreases (i.e., the number of elements increases), the model behavioral approximation should approach the closed-form mechanical solution. Although the individual elements may not describe the behavior of the whole very well, their combined effects may. In order to predict object behavior, two requirements are imposed on the combination of finite elements: (1) displacements must be continuous with elements, and (2) displacements must be compatible between adjacent elements [Desai and Abel, 1972].

The FONM approach describes discrete objects; however, they are discrete on a much larger scale than the finite element approach. Although the consistency and compatibility requirements for behavioral process primitives are based on the requirements imposed on finite elements, each object represents what in the FEM would be considered a large set of elements. FONM representations utilize the notion of a finite element to describe sequences of behavior associated with object function. By enforcing continuity between behavioral processes in a sequence, individual behaviors may not describe the function very well (like the discrete FEM element), but the overall behavioral representation is a viable account of the observed function.

For example, the FONM gear object is represented as an object with a center, an axle, and a tooth region which spans its entire outer edge. In FEM, the gear would be represented as many smaller plate elements, as shown in Fig. 8.21 (re-illustrated from Fig. 2.35).

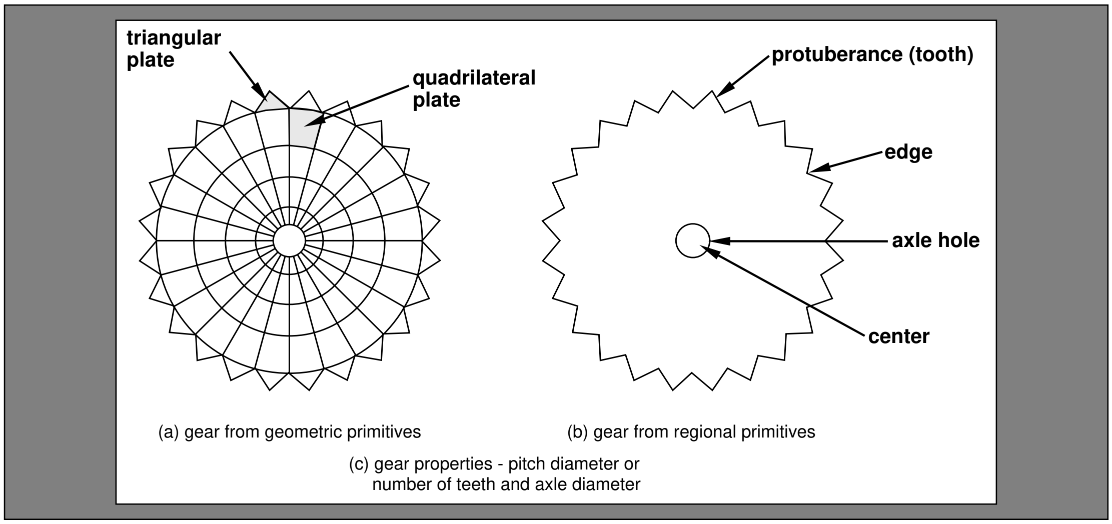


Figure 8.21 Comparison between FEM and FONM gear representations.

In FONM, because of the granularity of the object, only motion type and direction can be determined, whereas in the FEM gear internal stresses and displacements can be determined. Although the results are quite different, the approach is basically the same. Despite the difference in granularity, FONM dynamics is based on the assumption that applied forces and changes in position (FEM nodal displacements) will be continuous between connected objects. The FEM approach has an advantage in being able to predict the internal forces and moments with a high degree of accuracy, but suffers from size and complexity. The simple gear illustrated in Fig. 8.21a has 5 rows of 24 elements, or 120 elements. Each element has 3 or four nodes and 6 degrees of freedom per node, or 2,736 degrees of freedom for one gear. In contrast, the FONM gear in Fig. 8.21b has a single element, 4 regions (the edge is actually represented with two opposing regions), and 6 degrees of freedom per region, or 24 degrees of freedom. This representation size reduction (114:1) is a strength for FONM representations, supporting the representation of larger systems.

8.3 AI Systems for Reasoning about Objects

Two research areas of computational modeling have been particularly relevant in defining the scope of FONM and its demonstration models: (1) design and creativity, and (2) natural language understanding.

8.3.1 Design and Creativity Systems

One aspect of the design process is the effect of design constraints on the creation of an artifact once the constraints have been identified. Sriram proposed a calculus for describing and manipulating constraints in the ALL-RISE model, [Sriram and Maher, 1986]. ALL-RISE is an expert system for preliminary structural design. The system takes as input an architectural building plan, represented as an object (or frame), and the output is a ranked set of feasible alternative structural systems, also represented as objects. Constraint schema represent the causal dependencies between components and influences. There are five constraint types in the system: (1) synthesis constraints, which affect the construction of feasible configurations; (2) interaction constraints, which describe dependencies between components; (3) causal constraints, which describe the stress-strain equations on components; (4) parametric constraints, which describe requirements on component physical characteristics such as strength; and (5) evaluation constraints, which are used to rank alternatives. Constraints are represented as schema which are procedurally attached to structural components. ALL-RISE functions by performing a depth-first search through a hierarchy of predefined structural systems. A structural subsystem is selected and its constraint set is evaluated against the design criteria. When an interaction constraint is found, the affected subsystem is marked. When the current pass is completed, control shifts to evaluating the constraint on the other component, rather that waiting until all constraints on the first component are evaluated. The search is complete when all of the constraints, interaction and otherwise, have been evaluated for all subsystems. The result of the search is a set of systems which satisfy both the individual and combined structural constraints.

ALL-RISE and EDEXP are both based on behavioral object models, both implement objects and constraints as schema, and both utilize procedural attachment during simulation. ALL-RISE implements its behavioral relations as simple expressions, whereas EDEXP implements behavioral relations with frames. The design philosophies differ inasmuch as ALL-RISE identifies systems which meet a specific set of criteria (i.e., constraint satisfaction), while EDEXP identifies the constraints between structural subsystems (i.e., constraint formulation). Neither model implements a complete behavioral model.

JACK is a program which formulates hypotheses about hidden mechanisms within devices based on an input/output (i.e., functional) description of the device [Doyle 1989a, Doyle 1989b]. JACK is able to determine how devices such as a tire pressure gauge, a toaster, and a refrigerator function by generating all the potential combinations of mechanisms which will produce the observed behavior. JACK takes two inputs: (1) a description of a device's externally observable behavior, and (2) a set of mechanisms. The output is a set of plausible combinations of those mechanisms which explain the behavior. The problem is characterized as a graph, where the nodes correspond to device events and the arcs correspond to the mechanisms which produce state changes. The observable behavior represent the edges of the graph, and the solution is a connected graph between edges. Mechanisms are represented as combinations of 9 constraint types: (1) type constraint, which controls quantities; (2) delay constraint, which relates the occurrence of events; (3) sign constraint, which describes value direction; (4) direction constraint, which describes spatial value; (5) magnitude constraint, which defines values of physical quantities; (6) alignment constraint, which describes positioning requirements between components before and after an event; (7) bias constraint, which affects directions of change; (8) displacement constraint, which controls physical location of objects, and (9) medium constraint, which relates object connectivity. Each of these constraints can take on a value or values. Device events are also represented in terms of the same 9 constraints as mechanisms. There are three types of causal interactions in JACK: (1) enablement, (2) disablement, and (3) equilibrium. A mechanism is hypothesized from available events. Once enabled, the constraints associated with the mechanism are propagated to construct a new (i.e., expected) event. The representation in JACK is purely behavioral. There is no particular representation of device structure, nor of device function; only device states and means to produce them. The mechanism representation, being comprised of 9 constraints, has no classification scheme, nor any motivation for why the 9 constraint types were chosen. There is no way to compare two devices which use the same mechanism, or to substitute one device for another based on the mutual ability to effect a mechanism. For example, JACK cannot be used to construct a model for a light bulb switch, even though it can construct a model for a toaster or a refrigerator, which are each comprised of switch elements. This is because JACK is not defined by a set of object, behavior, or function primitives, as in FONM. One limitation this has on JACK's performance is that JACK must enumerate the possible mechanism combinations in order to find a plausible one. A knowledge-based approach, such as FONM, aids in reducing the combinatorics inherent in JACK by classifying behavior and function types and their dependencies. Although JACK isn't capable of determining which of the potential functional descriptions is the correct one, the model is interesting because many non-intuitive functional models may be proposed. JACK can be considered a creativity model, because it could potentially be given the behavioral description for a device that one *wants* to build and then suggest functional models which might aid the designer in implementing the device.

KRITIK is a problem-solving system which designs devices using design knowledge represented as cases and organized in an episodic memory of cases [Goel 1989]. Devices in KRITIK are based on the Functional Representation (structure-behavior-function, or SBF) approach. Cases are represented by combining the models associated with different representation levels (e.g., functional). Similarities in devices are represented in the cases by device functional specifications. KRITIK takes as input a specification of the constraints on a design, and produces as output a structural specification that satisfies the constraints. The problem-solving approach in KRITIK is based on three subtasks: (1) retrieval, (2) adaptation, and (3) storage. The retrieval task matches the design specification to cases in memory and retrieves the most similar one. The adaptation task modifies the retrieved design to satisfy the design constraints. The storage task stores the current design task in memory for future problem-solving scenarios. KRITIK uses an integrated approach to indexing cases for retrieval, based on the goals which are achieved by the design and by the features which are relevant to the goal. The KRITIK system is a useful experiment in design scenarios in which the design constraints are known, and in which successful design cases are available. It is not intended to, nor can it support general problem solving, for two reasons. First, the representation approach, being based on the intended, or expected, functions of devices does not allow for a device to be used outside of its intended context. The device can only be applied to a design scenario when the context is matched. Second, the representation accounts for all the structure-behavior-function device characteristics, but they are not represented as independent structures. Problem solving thus cannot adapt a plan for achieving a goal, with a device, rather than adapting the device to suit a plan. A general problem-solving model should be able to use the design case, or experience, in such a way that knowledge at any abstraction level can be applied to retrieval and adaptation of designs. Moreover, KRITIK doesn't appear to take into consideration the possibility of combining designs which partially satisfy the design constraints, rather then trying to adapt a single design to the task. The application of knowledge structures such as FONM device-use goals and machine primitives to the KRITIK model could enable it to perform its design task with the flexibility described above.

8.3.2 Natural Language Systems

RESEARCHER is an experimental model for reading, remembering, and generalizing patent abstracts in English [Lebowitz 1985, Wasserman and Lebowitz, 1983]. RESEARCHER represents objects in three ways: (1) by their component hierachies, (2) by the physical and functional relations between parts, and (3) by the properties of parts. The model reads and understands device components and their behaviors. An example of the kind of text that RESEARCHER reads is shown below:

> A disc head supporting a spindle made of magnetic material.

When the description is parsed, information about its components and their physical characteristics, their composition, and of their behavior is resident in memory. RESEARCHER disambiguates the text by using memory access. For example, RESEARCHER must access knowledge about "support" in order know which object is made of magnetic material. Otherwise the reference to composition cannot be disambiguated.

TAILOR is a natural language generation program which has been designed to generate descriptions of the devices analyzed by RESEARCHER. TAILOR is based on the premise that a users knowledge level is important in answering questions they might have about an object description, and that a generation model should be able to choose the right level to answer a question based on knowledge about the user [Paris 1987, Paris 1989]. TAILOR uses information about a user's level of expertise to select and combine discourse strategies for replying to queries about an object. TAILOR uses two discourse strategies: (1) constituency schema, and (2) process trace. Constituency is used when the user is expert about a particular aspect of an object, and then presents detailed information about the object and its characteristics when describing the object. Process trace is used when the user is naive about a particular aspect of an object, and presents a behavioral account of how an object works. For example, consider the text below, which is a description of a telephone generated by TAILOR based on the user knowing how a loudspeaker works. TAILOR selects the constituency strategy for most of the description, but, because the user doesn't know about transmitters, switches to a process trace for that component:

> The telephone changes soundwaves into soundwaves. The telephone has a various-shaped housing, a transmitter that changes soundwaves into current, a curly-shaped cord, a line, a receiver to change current into soundwaves, and a dialing_mechanism. The transmitter is a microphone. A person speaking into the microphone causes the soundwaves to hit the diaphram of the microphone. The soundwaves hitting the diaphram causes the diaphram to vibrate. The diaphram vibrating causes the current to vary. The current varies like the the intensity varies. The receiver is a loudspeaker with a small aluminum diaphram...

The textual component associated with TAILOR's application of the process trace is highlighted. It follows the portion of the description where the device components are enumerated.

TAILOR selects which strategy to apply based on what it knows about the user's understanding of the object. The user is deemed naive when he/she does not know about the basic concepts associated with the domain, such as how an engine works when describing a tune-up procedure.

RESEARCHER and EDCA are both models which represent knowledge at the structural, compositional, and behavioral levels. RESEARCHER has been extended well beyond what EDCA can currently analyze, because its behavioral model includes more domains than the purely mechanical domain used in descriptions parsed by EDCA. Neither system analyzes devices based on a functional level which is separate from the behavioral level, let alone about object use and experience. The FONM knowledge constructs support representation and description of device function and use, but have not been implemented in that capacity. The TAILOR model points out the need to describe objects at multiple abstraction levels and demonstrates that need with description of objects, such as a telephone, which are pertinent to different levels of expertise. Often a person's level of expertise is lower, even, than the causal, compositional, and structural levels which TAILOR has access to from RESEARCHER. In such cases the discourse strategy should be extended to support descriptions which make use of FONM machine primitive-like representations and device-use goals.

Granacki and Parker, and, more recently, Arens have developed a natural language front end, PHRAN-SPAN for their Advanced Design And Manufacturing (ADAM) system [Granacki and Parker, 1986, Granacki et al, 1989]. Object descriptions are based on a modified Conceptual Dependency representation model, and the domain is digital systems design. For example, PHRAN-SPAN reads the following sentence:

> The cpu transfers the block of data bytes from the disk to the control store.

To parse and understand this sentence, a Design Data Structure (DDS) was introduced to represent design data. The DDS has four components: (1) data flow, which represent data dependencies; (2) timing; (3) structural, which represents the composition of a circuit; and (4) physical, which represents the components and their properties. In addition, five conceptual structures are introduced to represent domain-specific knowledge: (1) information-transfer, (2) temporal-activities, (3) temporal-constraints, (4) control, and (5) declarations. For example, to parse the sentence above, a conceptual structure called VTRANS was proposed:

```
(uni_dir_vtrans
  (source  (a_component *unspecified*))
  (sink    (a_component *unspecified*))
  (info    (df_val command1))
  (control (a_component cpu1)))
```

The `a_component` fillers are unspecified when the concept is initially encountered, and are filled during the processing task. This structure is similar to the FONM BPP, although it is not apparent whether the ADAM project has identified a taxonomy of DDS primitives specific to digital design.

PHRAN reads and analyzes sentences one at a time, and word by word [Arens 1986]. The model represents words as pattern-concept pairs, where the patterns can be phrasal constructs, literals, or syntactic. A word is read, looked up in the lexicon, matched and then retained or eliminated from further consideration. The pattern match can be lexical, semantic, or syntactic. The knowledge domain in which PHRAN-SPAN is used, digital design, is much more dependent on timing relations than the domain of simple mechanical devices. As such, the temporal model used much more extensive than that in FONM. Both models focus on constructing explanations of descriptions which have elements of static and dynamic relations between components. Computationally, the models differ inasmuch as EDCA is a demon parser and PHRAN is a phrasal parser.

Recent work has extended the behavioral approach to language comprehension modeled in RESEARCHER/TAILOR, PHRAN-SPAN, and EDCA to incorporate functional modeling from the KRITIK system. The KA model integrates the acquisition, understanding, and analysis of text design cases using a Functional Representation approach [Pittges et al, 1993]. KA addresses the issues of ambiguity, underspecification, and information relevance at the functional, rather than behavioral, representation level. KA is being designed to take design specifications in English (like PHRAN-SPAN) and to produce a design which meets the specification. The underlying representation uses a structure-behavior-function (SBF) model that specifies how the structure achieves its function (where function is represented as the device purpose). The figure below illustrates the kind of text the KA model is supposed to analyze: One problem with this text is that no reference is made to a power source; KA must infer that a circuit has a power source or it cannot function proper-

> Consider a flashlight circuit. The function of the circuit is to produce light. The input is a small force on the switch. The output is light of eighteen lumens intensity and blue color.

ly, and so the design must have a power source. KA must also recognize that an "input" generally refers to a function’s initialization, that the force is being applied to the switch, and that switch movement results in the closed circuit since no other description, other than the output specification, is provided in the description. Although KA, when combined with KRITIK, is a fully integrated approach to conceptual analysis, it fails at being comprehensive in the same ways that RESEARCHER and EDCA fail, but from a different perspective. As pointed out in the TAILOR system, descriptions of an object must be available at any level of abstraction, and KA only provides a function-level description of the object. By performing the type of analysis that EDCA would perform with access to FONM machine primitives, KA extends the comprehension of objects.

○
○
○
○


PULL
MIDVALE
SCHOOL FOR
THE GIFTED

○
○
○
○

Chapter 9

Limitations of FONM

This chapter describes short term (minor) and long term (major) limitations of FONM and the EDEXP and EDCA demonstration models.

9.1 Representational Limitations

FONM is comprised of three levels of abstraction, each of which has a particular set of assumptions which define its scope and its limitations. The following sections address the limitations of FONM (1) statics, (2) dynamics, and (3) pragmatics representations.

9.1.1 Limitations of FONM Statics

There are six basic limitations to the representation of device statics in FONM:

- Spatial and temporal knowledge are not represented in detail
- Material and behavioral properties chosen are not exhaustive
- No support for property derivatives such as velocity and acceleration
- Complex shapes cannot be represented with regions
- Geometric primitive classification is redundant
- Object primitives constrain behavioral reasoning

There is no particular representation of space in the model. An adhoc symbolic method for describing relative position and value based on scales was used for the purposes of parsing representations into illustrations; however, this method was not intended to be robust because spatial analysis was not intended to be the focus of attention. The same approach was used to represent dimension and direction. As a result, FONM static object representations cannot represent position or direction well enough to perform detailed spatial simulations. The same limitations are true to a greater extent with temporal knowledge. When an event actually occurs, and its duration, are not modeled in any way in FONM, so the only temporal inferences which are supported are those which are either explicitly sequential or explicitly simultaneous.

The material properties chosen to represent objects are not claimed to be an exhaustive set of properties, but to represent the types of properties which can be used to support some behavioral inferences about what an object can do. For example, the stiffness of an object helps to determine whether it can transmit force. But the material properties do not support any notion of state change, such as from ice to water to steam, or the tensioning of rope by twisting, which dramatically affect the mechanical properties of an object. For example, there is no mechanism in the FONM approach for the idea that, if one twists rope a lot, it has some capacity to transmit forces in the `NEG` direction (i.e., one can push on rope if it is wound tightly enough).

The behavioral properties which have been selected to describe object behavior (position, restraint, applied force, internal force, and size) support the recognition and prediction of simple mechanical behaviors. They do not represent an exhaustive list of behavioral properties for all physical domains, nor even in the mechanical domain. For example, the notion of internal force is used as an abstraction to describe energy absorption, which is not explicitly true. As an approximation it is adequate at the representation level being described because, regardless of its representation as force or energy its effect, both in FONM dynamics and physically, is the same.

A fundamental limitation of the FONM object representations is lack of support for property derivatives. The qualitative reasoning community (e.g., Forbus, DeKleer and Brown, Kuipers, etc.) have developed calculi for symbolically representing velocity, acceleration, momentum, conservation of energy and mass, and so on which are not represented with FONM. This limitation makes it impossible to use FONM for any transient analysis. Since much of dynamics is associated with transient behavior, particularly for diagnosing machine and system faults, this limitation is problematic. All of these limitations are based on the granularity of the FONM statics. Each of these problems could be resolved by extending the scope of the static object model, because the dependencies between object statics, dynamics, and pragmatics are distinct in FONM.

The FONM region taxonomy was chosen to simplify the representation of shape and the number of locations on an object that needed to be explicitly represented. As such, the size of representations is reduced, but the type of simulations that can be performed using this kind of spatial model is compromised. For example, a FONM gear object has only four regions: a center (or rotation), an axle, and a tooth region on the outer edge (actually represented with two opposing regions for consistency with cylinder, wheel, etc.). Representing the gear with four regions enables representation of force transmission and magnification, direction of motion, and relative magnitude. It does not enable the representation of position along the edge, since the entire edge is represented with a single region. As a result, the FONM gear object cannot be used to simulate a timing sequence in a clock beyond half gear revolutions, because position of one tooth to another is not represented. The FONM object can support diagnosis of malfunction in a clock. Because gear contact is required for force transmission, it could be inferred that malfunction at the gear level was caused by a problem with the tooth region.

Regions also do not directly support the representation of truly complex shapes. For example, the region taxonomy allows the representation of a nutcracker jaw as a tooth region, where each tooth is represented as a wedge-shaped protuberance. Each protuberance has the capacity to restrain motion, and the wedge shape supports the possibility of cutting, but there is currently no mechanism to describe many teeth in a jaw or many teeth on a gear. The model simply makes the assumption that the entire region has the same behavioral properties as a single entity (in this case a tooth), and so the area encompassed by the region need not be represented explicitly in any other capacity.

The FONM object classification is a superset of volumetric primitives, selected to provide a basic set of uncomplicated shaped objects. The representation of the geometric primitives is limited to four characteristics: (1) center of gravity, (2) observable regions, (3) relative sizes in different dimensions, and (4) shape viewed along dimensional axes. The taxonomy of geometric primitives chosen is clearly redundant, since there are a number of ways to describe the same object and many of the objects are specializations of other objects. The filament and film, for example, are specializations of column and plate, which, in turn, are specializations of cylinder and block, which are true volumetric primitives. The limitations in this approach are the number of the objects in the set and how that number affects processing performance, though no specific processing based on geometric primitives is currently done in models which use FONM representations.

The notion of object primitive is used to describe an object which instantiates a machine primitive, and is not intended to take on a particular shape. The appearance of the object primitives was chosen to identify the required regions of the object, rather than its shape. By representing object primitives only in terms of those regions required to describe the FONM behavioral primitives and machine primitives, the approach is limited in the types of behavior which can be simulated. For example, a gear can be instantiated in a MOTION process, but since all gear teeth are represented as a single region of the object primitive, the inferences which can be drawn about a particular gear tooth are limited to the direction of motion with respect to the center and with respect to another point on the other side of the center. Detailed simulations are not supported by this description of objects. On the other hand, this level of description eliminates large amounts of information which are not required for a functional level of analysis.

9.1.2 Limitations in FONM Dynamics

The FONM notion of object dynamics is based on the assumption that it is important from an object use point of view to know how a behavior is enabled, and what type of effect it produces, but not necessarily how it is produced or to support its simulation. This assumption is based on the cognitive work of McCloskey [McCloskey 1983] and DiSessa [DiSessa 1983], which have shown that naive reasoners are able to recognize low-level behavior but are not able to predict low-level object behavior. Within this scope, FONM dynamic representations suffer from limitations of six kinds:

- BPPs do not describe all dynamic behavior
- BPPs granularity is coarse
- Complex behaviors not represented
- MPs are not true dynamic primitives

The motion-restrain-transform-store-deform BPP taxonomy is only valid for the intended scope of the representation and is not, otherwise, complete. For example, it is not possible to represent the momentum of objects using FONM, even though force, mass, and position can be represented (to some extent). As such, object collisions cannot be effectively modeled using FONM dynamics. There are two reasons. First, position rates of change are not directly represented. Second, the interaction of mass and other properties has not been addressed in FONM. The recognition that an object is in motion implies that the object has a velocity, so it must have a momentum, but FONM has no structure which represents it. One could claim that, at some level, momentum has the same type of effect as internal stored energy. Then it could be represented as an internal force and its behavior could be represented with the STORE process. This would require a rethinking of the role of STORE and its interactions with the TRANSFORM and DEFORM processes, and has not been attempted in this research.

The claim of primitives in FONM is based only on mechanical behavior, and not on behavior in other physical domains, despite the fact that many mechanical behaviors have analogs in other physical domains, and despite the fact that mechanical devices are often used in conjunction with other physical domains. For example, the RESTRAIN process is used to represent interactions which affect object motion mechanically, and could potentially be used to represent resistance in electrical circuits and viscosity in fluid mechanics. Likewise, the STORE process is used to represent internal energy in mechanical objects, and could potentially be used to represent capacitance in electrical circuits. In order to make claims about the general applicability of FONM behavioral primitives, each physical domain would have to have 5 behavioral classes which were analogs with the 5 processes in FONM, and their structures would have to be the same as the structure of behavioral processes proposed in FONM.

This 'level of representation' assumption, which defines the scope of low-level dynamics in FONM, also limits the applicability of FONM representations in mechanical diagnosis and simulation. The FONM representation of behavioral process identifies those aspects of a device's static representation which either enable a form of behavior or are changed as a result of that behavior, but the behavioral model does not specify how much change is produced nor does it represent the actual change. Thus, the FONM MOTION process can be recognized as being enabled, but it will only be possible to determine what kind of motion the object is engaged in, what the dimension and direction of its motion are, and the type of path it is on.

The behavioral primitives proposed in FONM have specialization hierarchies (e.g., Figs. 3.6, 3.11) which illustrate exemplars for their class of mechanical behavior. Examples which have been presented in this dissertation describe the basic specializations of the class but not complex behaviors which can be represented by combining them. For example, the FONM representation for BPP-MOTION was presented with basic specializations for translation, falling, sliding, rotation, rolling, and swinging. It is also possible to combine these motion types to describe complex motions, such as bouncing, which is a combination of falling, translating, and spinning. This type of combination has not been performed in this research.

Machine primitives represent primitive object/function relationships. Their abstraction layer in FONM enables access to how devices are used in a plan, and to how devices behave. They are not true representation primitives because they do not form the lowest inference level in FONM, which is represented with processes. The taxonomy of machine primitives has been chosen to encompass the true machines (lever and inclined plane), the primitive mechanisms as defined by the U.S. Navy [Navy 1972], and those which have been motivated by adhoc protocol studies. There is no evidence to support the particular choice of machine primitives, nor their organization in Fig. 4.4.

FONM dynamic representations can make the task of identifying role associations between objects difficult when the objects have a similar functional capacity but are applied in different ways than their static representations would suggest. Consider a broken fan belt, which is used as a component of the pulley-object, a confinement device. It has been suggested in [Rubinstein 1986] that a silk tie might be used for a makeshift fan belt, because of its elasticity, size, and shape. The tie ends would have to be tied, to form a belt of the right size, and the knot would have to be very tight (i.e., small) to fit the pulley channel. If a reasoning model makes comparisons between devices at the function level, then there is a possibility that the tie will be considered a potential replacement because of its role as confinement device. The tie confines the shirt collar and neck around which it is tied. But there is no mapping of regions between the fan belt and the tie when they are applied in this manner, so the tie would not be considered further. If the function level is bypassed when making comparisons, then a comparison could be made at the static level, and the tie could be tried on that basis, but so could many other objects which are not appropriate to the situation. In FONM, device-use plans are more closely associated with device functions than with device structure, so a model which made use of FONM representations might not be able to make the correspondence between the fan belt and the tie.

9.1.3 Limitations in FONM Pragmatics

The FONM representation of device pragmatics does not address two important issues which must be addressed by a problem-solving model:

- Device malfunction and misuse
- Property measurement in problem solving

Currently there is no representation for damage (such as a broken gear tooth) in FONM, only disabled behavior. If two gears must be in gear contact in order to function, and the gear function fails, then one of the inferences that can be made from a FONM representation is that gear to gear tooth contact has been disabled. Since a protuberance is required for gear contact, it could be inferred that the protuberance no longer exists, but no such inference is supported in any of the demonstration models which have been constructed in this research, nor is there an explicit representation for the broken tooth. This is a limitation to FONM pragmatics because most problem solving situations provide the problem solver with perceptual information about the objects on hand, which must then be interpreted with respect to knowledge about how they should function/behave. This type of analysis cannot be performed unless the representation contains information about possible failure modes in terms of geometry changes to the objects.

Property Measurement and Problem Solving

A problem solver may know that to open a varnish can, the prying object chosen must fit the can slot. He can satisfy this requirement by: (1) measuring each object before trying it, (2) trying only objects which he already knows will fit the slot, or (3) trying an object and see if it fits. He is more likely to try using the object first, than to measure its size or the sizes of every potential pry object, simply because in trying the object he is effectively measuring it. Naive problem solving is based on experience and experimentation. An object's property values play a greater role in planning failure and learning than they do in predicting whether the object is suitable for a task. The construction of problem-solving models requires a representation which associates the process of experimentation with an object, its function, and the goals which its application either achieves or fails to achieve.

9.2 Limitations of the EDEXP Demonstration Model

The EDEXP program simulates simple door mutation and creation through the relocation of a hinge, recognizes whether new doors can rotate, and constructs a rule for locating hinges that controls door rotation. The model has four limitations:

- limited application of planner
- limited use of mutation and problem-solving heuristics
- only static graphics were implemented
- limited device and behavioral representations

Although EDEXP was designed with a general recursive descent planner and problem solver, the demonstrator only addressed a single type of goal, used only one plan (MUTATE), and one metric (MOVE COMPONENT). Replanning always used the same metric, so the demonstration model never took into account the effect of moving more than one hinge, or moving another component (like the doorknob). The hinge was not allowed to be moved to a location on the doorslab at all, which would have been interesting. Selecting another object (e.g., the doorknob), or another metric (e.g., increase the size of a hinge, or change the metal hinge into a rubber hinge) would also have been interesting. The plans are not implemented declaratively, so the planner executes them procedurally when their use is indicated.

The original intent of EDEXP was to devise a computational model capable of retrieving, combining, and modifying device representations toward specific design and invention goals (e.g., design a door for dog entry and exit). Except for the MOVE COMPONENT metric used in EDEXP, problem-solving and invention heuristics identified in the EDISON project were not implemented.

A simulation is a much better aid to user comprehension when it has a graphical interface. EDEXP had an interface which translated the current device representation into a graphical display using icons. When a new device was created by the program, the illustration changed, so that the user could see what was going on and compare it to the demons which were firing during the simulation. The EDEXP graphical interface was not a dynamic simulation. It simply displayed the current device, and the interface was never extended to graphically display behavioral or functional simulations. Recently a number of graphics packages which simulate device behavior have been developed (e.g., Interactive Physics). The innovative idea is the object-oriented approach to constructing the device and having them interact properly. Although these packages are based on quantitative, rather than qualitative, behavioral models, they underscore the usefulness of graphics in aiding users comprehension of device mechanics.

EDEXP would never have been developed if a general model of all the knowledge needed in mechanical experimentation was necessary before constructing a model. Only those structures which were directly applicable to reasoning about door function were described in detail, while others were simplified to account for their later inclusion in FONM. For example, in EDEXP the basic notions of motion and restraint are fairly well defined, but the structure and taxonomy of behavioral primitives were not yet fully developed. The notions of goals and plans as device pragmatics, and their role in the experimentation process, were not yet developed at all, even though the planner was based on goals and plans.

9.3 Limitations of the EDCA Demonstration Model

The EDCA program parses English descriptions of mechanical device behaviors underlying the functions of a toy dart gun, a press, and a door. The model has four limitations:

- No object statics or machine primitives are used
- No means for disambiguating independent multiple object references
- No reasoning support
- No question/answering or generation
- No learning support

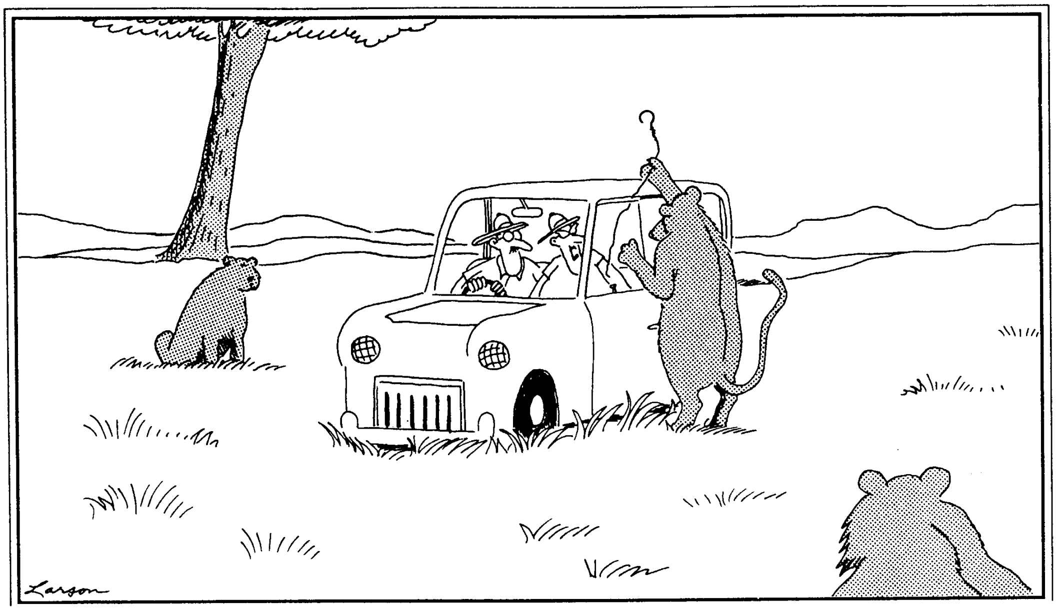
The EDCA model implements the FONM behavioral primitives, but neither the object static descriptions nor the machine primitives. The model can infer the presence of a behavioral primitive by finding the states which enable it or result from it, but not from the properties or regions of the associated objects or from recognizing an object and using its function to help infer its behavior. This is particularly evident in the parsing of Toy Gun, when the object reference for “it” cannot be disambiguated. With a memory which includes different toy dart guns, this reference could possibly be understood.

EDCA can disambiguate multiple references to a single object; however, if the same word is used for different objects, such as the gears in a transmission, or the screws used to connect a hinge to a door, then EDCA will erroneously associate all of them to a single object.

Although reasoning is used to construct the conceptual representation of the description, EDCA doesn’t support any reasoning about the device knowledge constructed by the conceptual analyzer. Questions cannot be asked and then traced through the representation in either direction.

There is no support for question-answering or generation in the EDCA implementation.

Although a primary use of a natural language model capable of reading and comprehending device descriptions would be to develop a device library from how-to books, no learning mechanism was ever implemented in EDCA.



"Drive, George, drive! This one's got a coat hanger!"

○
○
○
○

Chapter 10

Extensions, Applications and Conclusions

This chapter is organized as follows. First, the status of FONM and the implementation status of demonstration models are presented. Second, the future work necessary to extend the FONM approach to support EDISON project goals is discussed and possible approaches are presented. Third, potential applications based on FONM and its process models are discussed. Fourth, a summary of contributions and significance of this research are discussed.

10.1 Status of FONM and its Process Models

FONM independently supports the two processing tasks presented in Chapters 6, 7: (1) simple mechanical experimentation, [Dyer et al, 1986], (2) comprehension of three short behavioral descriptions [Dyer et al, 1987], (3) recognition of low-level MOTION, RESTRAIN, and TRANSFORM processes and their specializations [Hodges, 1988], and (4) hand coded solutions to three improvisation scenarios [Hodges, 1989, 1992a, Hodges et al, 1992b]. The approach currently supports a library of 19 device statics, dynamics, and pragmatics representations in four device classes. The parser lexicon currently consists of 75 entries for the three device descriptions currently parsed. There are 460 demons in the EDCA, EDEXP, and related models combined.

10.2 Future Directions

Full FONM device representations have not been implemented in a single process model which can perform the separate reasoning tasks illustrated in the EDEXP and EDCA demonstration programs, nor have the reasoning tasks associated with these models been significantly developed. The development of such a model would enable researchers to compare simulations using FONM representations to data gathered in experiments on human subjects. The results could be used to determine its suitability for representing the knowledge associated with naive mechanics. By constructing an integrated reasoning model, it could be determined to what extent the approach can be merged with lower level and higher level representation approaches, and to what extent the approach supports simulation of device behavior, function and use.

The first step toward constructing an integrated processing model would be to develop the demonstration programs into systems that utilize the full representation and perform the appropriate reasoning tasks. EDEXP could be extended to utilize both mutation and combination plans when experimenting, instead of the mutation plan alone. The mutation types used could be extended to include all the FONM state types (material, position, restraint, applied force, internal force, size and shape). The model would have to be updated to associate its experimentation with the behavioral and mechanical goals proposed in Chapter 5. Those goals in turn would have to be associated with higher level metrics for making choices between plans. For example, we developed a set of six invention metrics early in the EDISON project which were never implemented in EDEXP: (1) simplicity, (2) efficiency, (3) performance, (4) cost, (5) novelty, and (6) elegance. The application of these problem-solving metrics, combined with a planner with access to FONM device representations, and based on mechanical goals and plans would make a significant improvement in the program.

Currently no FONM-based model performs top-down simulation at either the behavioral or functional level. The EDEXP model could be extended by incorporating a perturbation and inference mechanism which would support top-down prediction of behavior. Using the same method in reverse, an explanation component could be developed.

EDCA could be extended by incorporating object statics, machine primitives, and device pragmatics into the lexicon and into the disambiguation and reasoning demons. These changes would enable EDCA to utilize both top-down and bottom-up reasoning during conceptual analysis. The efforts of Pittges et al to include functional reasoning capabilities into language comprehension models illustrates the need for this type of extension [Pittges et al, 1993], and the generation of device descriptions based on user expertise [Paris 1985, Paris 1987] illustrate the need to represent and reason about device descriptions over a broad spectrum, from very naive to expert. The FONM representation of function, and its representational distinction from goals and plans makes it a logical choice for this type of language extension. Another extension would be to develop a question-answering component to the system.

An improvisation model using FONM device pragmatics has never been implemented as a computational model, even though hand coded situations have been represented [Hodges 1989, Hodges 1992, Hodges et al, 1992]. The general idea is to analyze a situation by identifying potential goal failures and then associating them with affected plans and devices which can be used in those plans. An important capability for this type of program would be to utilize entire experiences organized in memory to look for similar goals, plans and devices, and to reincorporate the current scenario back into memory when problem solving is complete. The implementation of this model would require a memory model and a theory for assimilating new experiences into memory. I have done some preliminary work on organizing experiences in memory by the goals which are achieved or failed and by the devices which are used [Hodges 1989]. The model should also be capable of associating naive experimentation with device characteristics.

10.3 Potential Applications of FONM

A number of applications of FONM have been identified by constructing process models for each of the two processing tasks demonstrated: (1) experimentation and (2) conceptual analysis. The potential application of a representation approach which can support general reasoning about mechanical devices is limitless; however, a few specific applications have been identified below.

10.3.1 Experimentation and Invention Applications

Device experimentation could allow an intelligent system to learn about the behavior of new devices without assistance, or acquire knowledge through experimentation. Device invention could be used to rethink design scenarios and support the solution of problems remote to human access (e.g., Hubble telescope and Mars satellite problems). A design understanding and invention system capable of reasoning and brainstorming about physical devices, their functions and behavior, and capable of communicating that reasoning to a designer.

10.3.2 Simulation Applications

The simulation of object behavior has broad practical use for helping people to understand how objects function and behave. Two applications can readily be discerned: (1) machine diagnostic analysis, and (2) inferring and learning device mechanisms.

A mechanical reasoning system could analyze device behavior and function for arbitrarily complex machines. It could be used to simulate and diagnose behavior, or suggest ways to find out what is wrong with a disfunctional component or machine.

A system that has access to knowledge about mechanisms and the objects which effect them could be used to recognize the components and mechanisms of new objects from a description of their I/O behavior.

10.3.3 Language Processing Applications

Device comprehension could lead to the implementation of systems capable of communicating knowledge about device behavior and function to humans. Two examples are: (1) tutoring systems, and (2) database generation systems.

On-line tutors could replace written manuals. Manuals are difficult to read because they require the reader to supply necessary index keys, and manuals are only able to answer anticipated questions and rarely completely enough for naive reasoners. A system capable of behavioral and functional reasoning about devices could answer questions about devices and answer ‘why’ questions at a level understandable by the naive mechanic.

With the capability to understand natural language, a system could take descriptions from a designer or a text and interpret them (i.e., generate a conceptual representation). This representation could then be indexed into the conceptual database for later use, specifically for the types of applications already described.

10.3.4 Improvisation Applications

Improvisation is a capability well suited to people or machines placed in new and unfamiliar environments, or highly constrained situations. A system capable of interpreting object function based on context could be applied to application types requiring horizontal reasoning, such as conceptual design. A system that could reason about object use in context could be used to suggest ideas during design formulation. Given a particular status of a project, and a memory of other projects, the system could compare the new goals and constraints against earlier designs. This could lead to the resolution of design ‘ruts,’ and perhaps to new approaches to known problems.

10.4 Summary and Conclusions

FONM is a representation theory for describing simple mechanical devices and supporting the kinds of reasoning associated with naive mechanics. The approach to developing FONM representation structures was to: (1) identify the knowledge that is associated with day-to-day mechanical reasoning, (2) develop structures and processes which describe those knowledge types, and (3) show how those structures and processes are used in various tasks requiring mechanical knowledge.

To create a computer program that models a naive mechanic’s understanding of mechanical devices for use in problem-solving tasks, the program has to implement four knowledge theories:

- A theory of machines.
- A theory of function and behavior.
- A theory of device interpretation and use.
- A theory of reasoning and inference about device behavior, function, and use appropriate to the processing task.

This dissertation has presented a theory for representing mechanical device structure, behavior, function, and use. The FONM model of naive mechanics has addressed these requirements in six ways:

- **Abstraction levels**: 4 abstraction levels for representing mechanical devices have been identified.
- **Device statics**: Knowledge structures and taxonomies were introduced for representing device characteristics from which mechanical device components and configurations can be described.
- **Low-level device dynamics**: A behavioral process primitive knowledge structure and a taxonomy of 5 behavioral primitives were introduced for representing complex behavior and function.
- **High-level device dynamics**: A machine primitive knowledge structure and a taxonomy of 11 machine primitives were introduced for representing complex mechanical devices.
- **Device pragmatics**: Taxonomies for goals and plans which are specific to mechanical device use

and application were introduced.
- **Knowledge dependencies**: The knowledge dependencies which support reasoning between the representation levels were identified.

At each abstraction level, the knowledge necessary to describe an object and its causal interactions was identified, a canonical structure for representing the knowledge was developed, and a taxonomy of related knowledge structures was proposed. The usefulness of FONM representations was shown by constructing 2 experimental process models which demonstrate the application of these knowledge structures on reasoning tasks associated with mechanical problem solving. Although the community is divided on the definition of function, there is widespread agreement on the use of abstraction levels and which levels are useful in representing and applying device function (e.g., [Chittaro et al, 1993, Lind et al, 1993, Qian and Gero, 1993]).

Introduction of Device Regions

The introduction of device region and its taxonomy in Chapter 2 provides a mechanism for simplifying the geometric description of devices and for recognizing device function. Regions provide for a means to reduce the size and complexity of device representations because only those regions which are associated with machine primitives are identified in the device's physical description. Regions provide a means for reducing processing effort by providing a metric for describing object appearance along with object function and behavior. Researchers in the area of artificial vision are now beginning to use regions to help identify objects (e.g., [Rivlin et al, 1993, Stark and Bowyer, 1990]).

Introduction of BPPs and MPs

The notion of a behavioral process is not a new concept and has been used since the inception of naive physics [Hayes 1978, Forbus 1984, DeKleer and Brown, 1984, Kuipers 1984]. However, the FONM behavioral process primitive proposed in Chapter 3 captures a naive reasoning level while retaining much of the structure and content of existing approaches. In addition, FONM a taxonomy of behavioral primitives is proposed. The taxonomy results in an overall savings both representationally and computationally: the small number of structures which can represent a behavior means that there are fewer unique structures to keep track of, and the small number of primitives means that there are fewer unique inferences that can be made.

The notion of a function-based knowledge structure is relatively recent in qualitative reasoning (e.g., [Dyer et al, 1986, Sembugamoorthy and Chandrasekaran, 1986, Ulrich and Seering, 1987, Lind 1990]). In addition to the representational and computational benefits of behavioral primitives, Machine primitives provide a mechanism for constructing device models based on a skeleton which is causally related to, and providing access to, device use and behavior through the common device and its structure. The taxonomy of machine primitives proposed in Chapter 4 provides a finite set of structures from which to compose representations of mechanical device function.

Introduction of Device-related Goals and Plans

Other researchers have addressed the issue of how humans relate to objects and how objects are used to achieve goals [Lehnert 1978, Rieger 1978]. FONM contains a set of device-specific goals and plans (Chapter 5) which directly relate intention with device function and behavior. In each, a taxonomy of the related knowledge structure was introduced.

Combination of Object Use and Function

Previous approaches to representing and reasoning about devices have addressed the issue of interpreting object use and function; however, FONM represents relationships between device use and function, in Chapter 5, which are consistent with (a) how devices are used by humans, and (b) how devices behave and function. Recently there has been interest in this level of reasoning within the functional modeling and design community, with an emphasis on identifying how device function relates to how the device is interpreted by its user (e.g., [Qian and Gero, 1993, Kannapan, 1993]).