COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 62, No. 2, September, pp. 147–163, 1995

Functional and Physical Object Characteristics and Object Recognition in Improvisation

JACK HODGES

Computer Science Department, San Francisco State University, San Francisco, California 94132

Received August 30, 1994; accepted May 12, 1995

Improvisation is a task which requires the problem solver to identify a situation, its constraints, and the capabilities of available objects. Since problem solvers rarely have specific information about the properties of the objects they see, they must somehow attribute "properties" to objects based on their experiences. In this paper, the relationship between the physical properties of an object, its functional representation, and its use in problem solving will be explored. A representation construct called a property attribute is used to represent functional properties attributed to an object in the context of successful functional application. Attributes can be used in problem-solving contexts to aid in object recognition. An overview of the Functional Ontology for Naive Mechanics (FONM) representation model and theoretical examples illustrating the application of attributes to object recognition and problem solving are presented. © 1995 Academic Press, Inc.

INTRODUCTION

When people have to resolve problems involving mechanical objects in real-life situations, they must make decisions based on conflicting goals and constraints at both the functional and behavioral level. Consider the following improvisation scenario where these differences lead to a goal success:

Hanger

A woman's car breaks down in a seedy part of town. When her husband rents a tow dolly to come retrieve the car, they find that neither has brought the keys to the dysfunctional car. In an effort to reduce their interaction with locals, and to get out of the neighborhood quickly, the husband searches for a hanger equivalent and finds one in a bicycle pant leg protector. Soon the couple are on their way with the car loaded onto the dolly.

There are many representational issues in the *Hanger* scenario: (1) the relationship between the couple's goal to retrieve the vehicle and to preserve their personal security, the choice of plans under these circumstances; (2) the identification of plausible plans; (3) the identification of constraints on those plans; (4) the perceptual recognition of objects in the environment; (5) the recognition of functional and behavioral similarities between objects and the environment; (6) the adaptation of objects used in one environment to fit the constraints of the current environment; and so on. A primary issue from a computational perspective is the recognition of similar functional capabilities of objects which have never been used in the same context. In such cases, the problem solver must rely on knowledge of what contexts an object has been applied in, toward what end, and whether the application was successful. This evaluation process involves a representation of experience based on function [5, 8, 9], but it also requires a representation of functional capacity which is not specifically related to physical or visual object properties, but, rather, on the knowledge that the object has been successfully applied in other problem-solving situations.

Consider the scenario presented in *Hanger.* One plan which the couple might apply is to find a way to gain access to the car's interior, so that they can put the car into a neutral gear for towing. One such plan that many of us have had to employ is to find a hanger, bend it into an appropriate shape, slide it between the car door and window, and fiddle around with it to unlatch a door or window lock. What makes a hanger applicable for such a plan is its combination of physical properties: length, diameter, stiffness, and ductility. These exact property values, however, are not too important to a problem solver, despite their import to the problem.

More than likely, what the problem solver remembers is that a hanger works in this sort of context. What s/he seeks, when a hanger cannot be readily found, is an object which exhibits "hanger-ness." Unfortunately, such a property does not exist, any more than "metallic," or "wooly," or, for that matter, "long." Each of these characteristics is actually a property *ascribed* or *attributed* to an object, in relation to other objects, based on the object's functional capacity in a specific context or family of contexts. The attributed property is necessarily associated with the real properties of the object, but the problem solver may not know what the actual properties are, let alone their values or value ranges. Much of the complexity associated with

1077-3142/95 $12.00

recognizing an object, or recognizing its capabilities, is based on geometric characteristics of the object and its surroundings. Thus, a formalization of geometric characteristics that is based on what an object has been used to accomplish or might be used to accomplish should simplify the processes of recognition and interpretation.

In the remainder of this paper, a review of the Functional Ontology for Naive Mechanics (FONM) representation theory will be presented, identifying general causal relationships between device structure, behavior, function, and use. This review will be followed with a discussion of how object characteristics (particularly regions and attributes) are used in FONM to assist in recognizing and applying object behavior, function, and use, as needed in creative problem-solving scenarios like *Hanger*.

FONM AND DEVICE REPRESENTATION

FONM is an integrated theory for representing device function both in terms of its low-level structure and behavior and in terms of its use in problem-solving contexts. FONM is based on the use of causal primitives for representing behavior and function, upon which higher levels of causality are abstracted. The approach is composed of three interdependent abstraction layers: (1) device statics, (2) device dynamics, and (3) device pragmatics, as shown in Fig. 1.

A static device representation is one that describes the device at rest. A dynamic device representation describes what will happen to a device when perturbed, precluding transient effects. A pragmatic device representation describes how and why the device is used.

Device Statics

Object structure is represented as a collection of states, regions, relationships, and processes. States describe unary property values (geometric, material, and kinematic) a device takes on from moment to moment, while regions represent locations on an object which are pertinent for describing behavior and function. Regions are an important aspect of the theory because they represent the appearance of an object by how a generalized area can behave. Regions have four roles: object, shape, size, and location. The region hierarchy used in FONM is depicted in Fig. 2.

Since function is an observable phenomenon, regions are visual function identifiers. Regions describe surfaces and surface intersections on an object and represent functional volumes. For example, a protuberance region (at **1** in Fig. 2) is a convex-shaped surface and represents a volume of interference, while an indentation region (at **2**) is a concave-shaped surface and represents a volume of containment. Cracks and holes are functional volumes which have previously been problematic to represent because they describe the absence of material. It should be noted that both indentations and protuberances represent relationships beween visible features, which, in general, describe length and width, rather than depth. Since an indentation region represents a concave surface and is associated with containment/confinement, a hole can be represented as an indentation whose depth is the size of the object (i.e., NOM, for nominal). Since motion is based on unbalanced force and unobstructed passage, an indentation's capacity to support motion need not be an explicit representation of depth. If the hole is the size of the material, then motion through the material is supported. The notions of indentation and protuberance are thus general in that they enable the representation of confinement and containment with the same structure, where only the boundary surface is pertinent to the range of motion. In addition to an object's physical characteristics, FONM also represents spatial dependencies between objects by their relative orientation and placement. Also, at the structure level, processes represent connectivity and stored energy.

Objects fall into one of two basic categories in FONM: (1) those which transmit force in at least one dimension and direction—called *linkages*, and (2) those which do not—called *stuff*. Within the linkage designation, a set of geometric primitives is used to ground the representation of shape. A set of object primitives (e.g., linkage-object, lever-object, and spring-object) is abstracted from this set. Object primitives loosely describe objects which can instantiate machine primitives (a device dynamics construct). An overall static representation is composed of the combination of these constructs. For example, a simple articulated device, a nutcracker, is depicted as we normally see it in Fig. 3, followed by its idealization in FONM object primitives in Fig. 4, and by a FONM static device diagram in Fig. 5.

Each object primitive is represented by its regions and has an "idealized" icon which is used to display object configurations, such as the four icons used to depict the idealized nutcracker in Fig. 4. The idealized icon is used for consistency in depicting representations, whereas the primitive object, itself, is geometrically flexible. For example, a primitive spring object is always illustrated as a coil spring, but the FONM representation for the spring primitive object makes no geometric requirements for springs, so that a helical spring, a leaf spring, or a flat spring can be used equally well in similar static representations. The functionally important aspects of the spring are the spring constant and in which dimension the retarding force is effected and not necessarily its shape or size. The regions labeled on the components in Fig. 4 are those of the associated primitive objects. For example, `end1-s` and `end2-s` represent the regions for the primitive spring object. The connectivity of these components is illustrated in a static device diagram (Fig. 5), where the connections would be represented with FONM restrain processes.

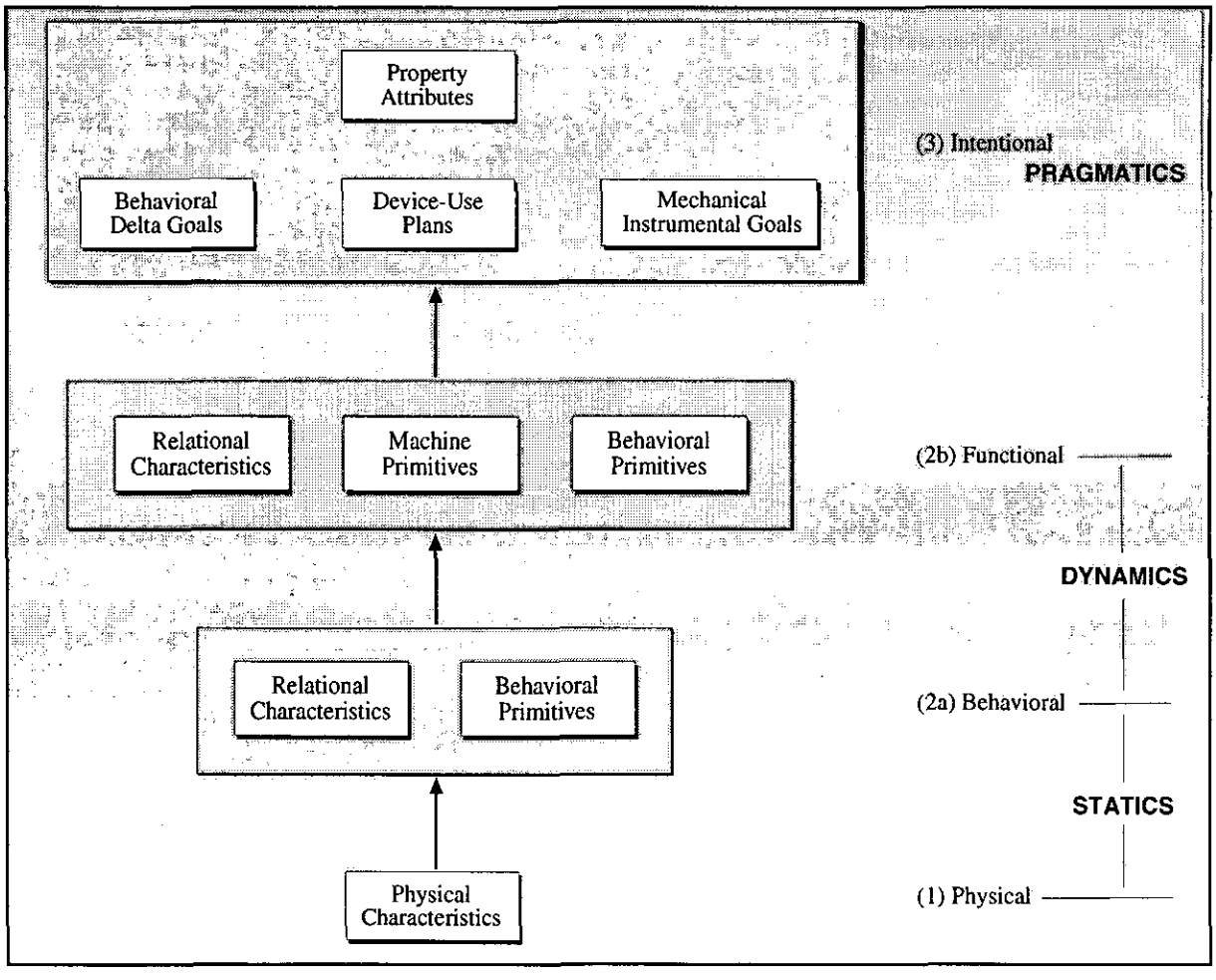

FIG. 1. Abstraction layers in FONM.

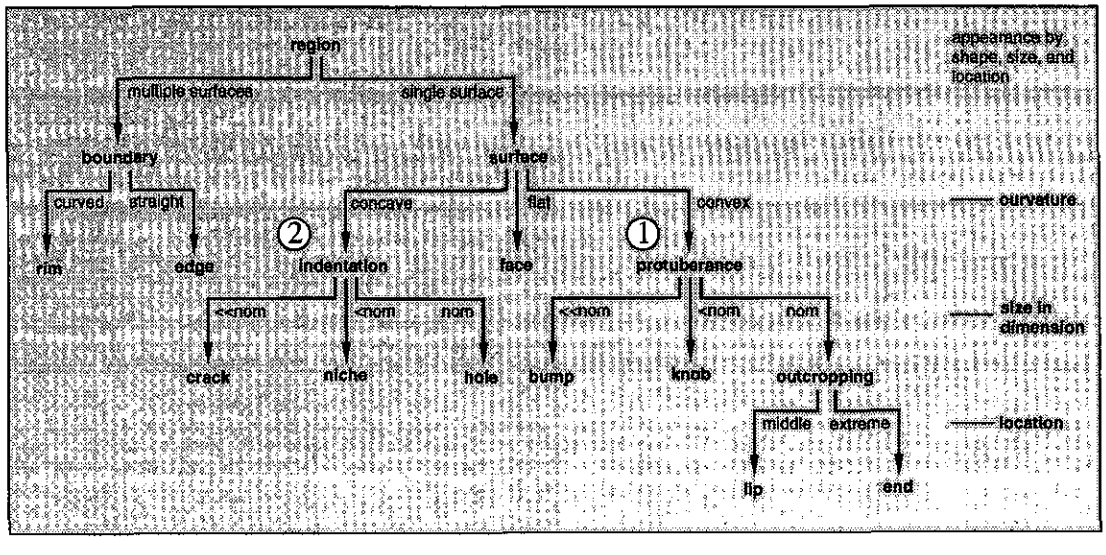

FIG. 2. FONM region hierarchy.

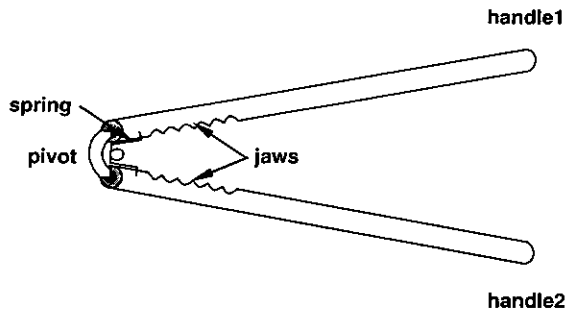

FIG. 3. A nutcracker.

The processes which represent these connections are not specifically illustrated in the figure, nor are the spatial relations (orientation and placement) between the components. The importance of the static object representation is that it describes the object before, between, or after dynamic events, and so defines the dynamic enablements and disablements which prescribe how the object will behave or what the object can be used to accomplish. For example, a nutcracker in its open state describes a spatial configuration of the components, and a state of the spring tension that defines the limit of handle motion in one direction and enables the cracking function.

Device Dynamics

Behavior is represented in FONM with a frame structure called a process (similar to Forbus' Qualitative Process [3], which describes the enablements and resulting states associated with mechanical device behaviors (Fig. 6).

The general process structure has six roles: for the process participating objects (*src* and *dst*), the applicable dimension (*dimr*), starting and ending states (*from* and *to*), and any process which is instrumental to this one (*inst*). A process' *from* and *to* states refer to a specific kinematic property, so that dynamics can be specifically inferred given a property type. A process takes place at an object region and can be enabled/disabled by any static representation component (i.e., geometric and material properties, and relations). There is a taxonomy of five *behavioral process primitives* in FONM: (1) motion, (2) restrain, (3) transform, (4) store, and (5) deform, as depicted in Fig. 7.

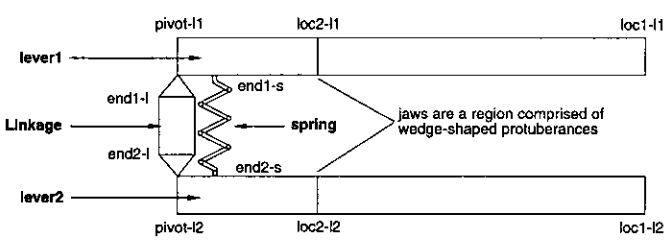

FIG. 4. An idealized nutcracker without jaws.

Motion represents the process which causes changes in an object's position with time. *Restrain* represents the process which causes changes or maintains the restraints on an object's degrees of freedom. *Transform* represents the process which causes the transformation of force between objects. *Store* represents the process which causes absorption or release of elastic energy (stored) in an object. *Deform* represents the process which causes permanent change of object shape and size associated with plastic deformation. The labels in Fig. 7 identify specific aspects of the representation. The item ("restraint," at **1**) identifies the kinematic property type that changes as a result of an enabled process, in this case *Restrain.* The causal relation ("enables," at **2**) illustrates a causal relation between position and restraint which enables force transformation. Notes such as that at **3** identify additional enablements on the associated process. Quoted words, such as "compress" (at **4**) identify words that exemplify an instantiation of the associated process. Since each behavioral process primitive is associated with a specific behavioral state, there is a 1:1 relationship between what behavior is involved and what kind of state change is involved. Behavioral process primitives can be combined into sequences to describe more abstract behaviors, similar to the way Schank combines action primitives into sequences called scripts and plans [10].

Some process sequences have observable starting and ending states which are associated with device function. One such sequence, which represents the lower level behavior of a nutcracker when a nut is being cracked, is shown in Fig. 8.

In this sequence, the two nutcracker handles are held somewhere along the length of the levers. This nonspecification of location is characteristic of a region description, which emphasizes what is important about the location rather than the location itself. The action of holding then causally enables the actor to press the handles together, imparting force to the handles. The combination of the applied force, the connection of the handles at their pivoted ends, and the presence of the nut between the jaws (again, a generalized location) enables mechanical advantage (`Transform-magnify`) at the jaw region. The magnified forces on the nut, and the disablement of nut motion between the jaws, enable a `Store-compress` process and, if the force is great enough, a `Deform-crack` process on the nutshell. The overall sequence represents the cracking function of the nutcracker.

Function is represented with a schema that describes the initiating and terminating states associated with a process sequence (Fig. 9). Function has three other, object-specific, roles that represent the locations where input/output occur: (1) an *appl* region where force is applied, (2) a *react* region where force is reacted, and (3) a *pivot* region where appropriate. When these locations are not instantiated with

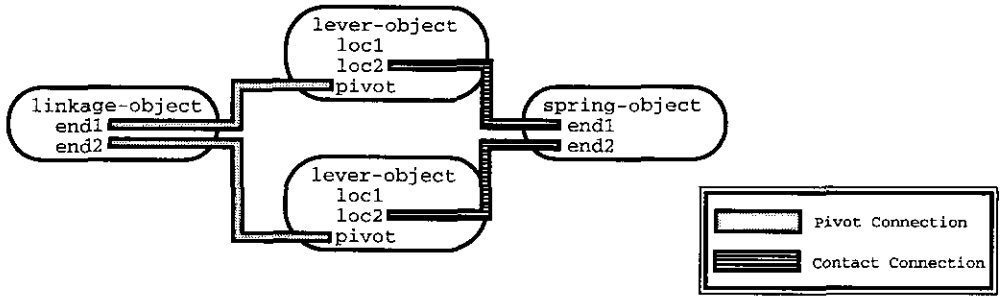

FIG. 5. Static device diagram for an idealized nutcracker.

object regions, the object's center of gravity is assumed. Each of the processes in the sequence describing the function must be enabled for the function to be enabled, requiring the function roles to fulfill the property and relational requirements at the lower (i.e., behavioral) abstraction level. In addition, function is initiated and terminated by (i.e., they instantiate the function *from* and *to* roles) perturbation states resulting from processes or actions. These states are generally the bounding states observed in device function, when such states are observable.

For example, a nutcracker has two bounding states: "open" and "closed" which can be associated with geometric maxima and minima for the device's range of motion. These states initiate and terminate the "reenable" and "crack" functions and disable each other as exclusive states (like on/off). An example illustrating this relationship for two functions (prying and pounding) of a screwdriver is illustrated in Fig. 10.

In this figure, function is depicted with an oval, and states, actions, and processes are depicted as before. The POUND-CLOSED function depicted on the left (10a) illustrates only the observable (black box) aspects of the function, while the PRY-OPEN function on the right (10b) is fleshed out to the processes which it instantiates. The bounding state of one function enables the other, as shown at **(1)** and **(2).**

Some functions are easily identified, either by what states they produce or by the type of object that they are generally associated with. For example, a linkage primitive object is associated with a function that transmits or translates force. The representation for this function is actually a single behavioral process, as shown in Fig. 11.

In this figure, the two states that enable and result from the force transmission (at **1** and **2**, respectively) are associated with the states that precede and follow the process sequence in Fig. 9 (also at **1** and **2**). The restraint state

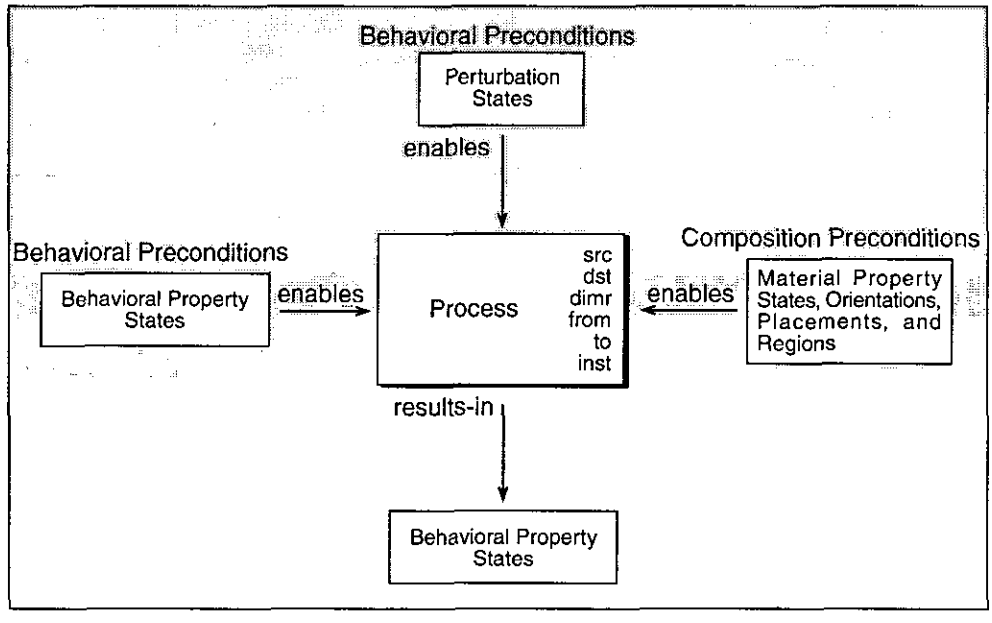

FIG. 6. Canonical structure for the FONM process and its state dependencies.

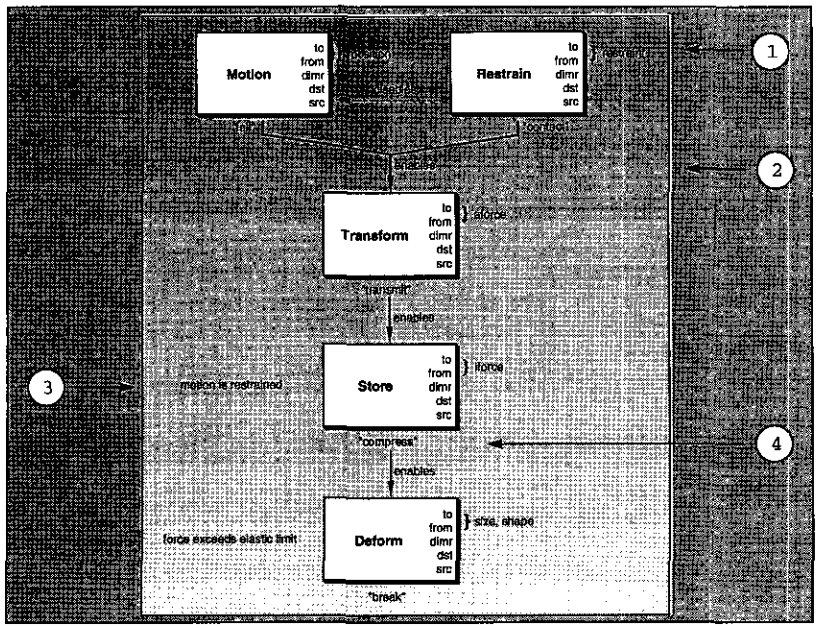

FIG. 7. FONM process taxonomy and causal dependencies.

resulting from the process at **(3)** in Fig. 11 is also pictured at item **(3)** in Fig. 9. As mentioned, this process sequence has a primary function associated with a single process (transmission), which is shown at **(4)** in both figures. In FONM, the function is considered a black box within which it does not really matter how the result is actually produced. The box surrounding the process `transform-transmit` is effectively that black box for the linkage function. This is shown again (as an *MP block* at **1**) in Fig. 12.

The states that initiate and terminate the function `MP-Linkage` are instantiated for the regions `appl` and `react`, which represent the regions where the force is applied and reacted. The `restrain-contact` process between the linkage-object and some other object enables the force transmission associated with `MP-Linkage`. `MP-Linkage` is called a *machine primitive*, because it is a fundamental, functional, building block that is associated with a single function. Like a behavioral process primitive, having a single function associated with a specific context enables explicit inferences when that context is instantiated. There is a taxonomy of 11 machine primitives in FONM: linkage, lever, wheel, gear, pulley, bearing, container, plane, blade, screw, and spring (Fig. 13).

Each primitive describes an object which has a single expected function. The representation prescribes the physical characteristics of the object (a primitive object) just enough to enable the associated function schema. As such, the representation of function is intentionally flexible with respect to object geometry/appearance. The items in Fig. 13 are illustrated with their role specializations from the function canonical form. For example, `MP-Linkage` makes use of only the `appl` and `react` roles of the function structure. When a pivot is added to the linage-object, then a new machine is described, a lever, which is represented with `MP-Lever`. `MP-Lever` makes use of all three object-specific function roles. When the lever-object is instantiated by a wheel-object, the function roles are specialized to edge, center, and axle, respectively. Each primitive inherits functionality from its parent. Thus, `MP-Bearing` inherits the functionality of `MP-Wheel-Axle`, `MP-Lever`, and `MP-Linkage`. Finally, the text below each primitive describes the type of functionality which the primitive produces when enabled, so `MP-Screw` is seen to produce rotational motion advantage as well as rotational mechanical advantage.

Like behavioral primitives, machine primitives can be combined into causal sequences, in this case to describe device functions. Whereas a static device description is based on components, a functional description is based on what the device component does. The requirements for combining machine primitives are that they maintain consistent forces and motions at their boundaries. The *MP-Diagram* (i.e., composed of MP Blocks) for the nutcracker cracking function is illustrated in Fig. 14.

Two functions are illustrated in this figure. The four primitives labeled `linkage1`, `pivot`, `linkage2`, and `spring` identify the components which instantiate the linkage-objects and the spring-object of the nutcracker at rest. In FONM this is called *a generalized linkage* because the object is effectively no more than a stick. When an object is placed between the two jaws, the region which instantiates the `linkage1` and `linkage2` `appl` roles

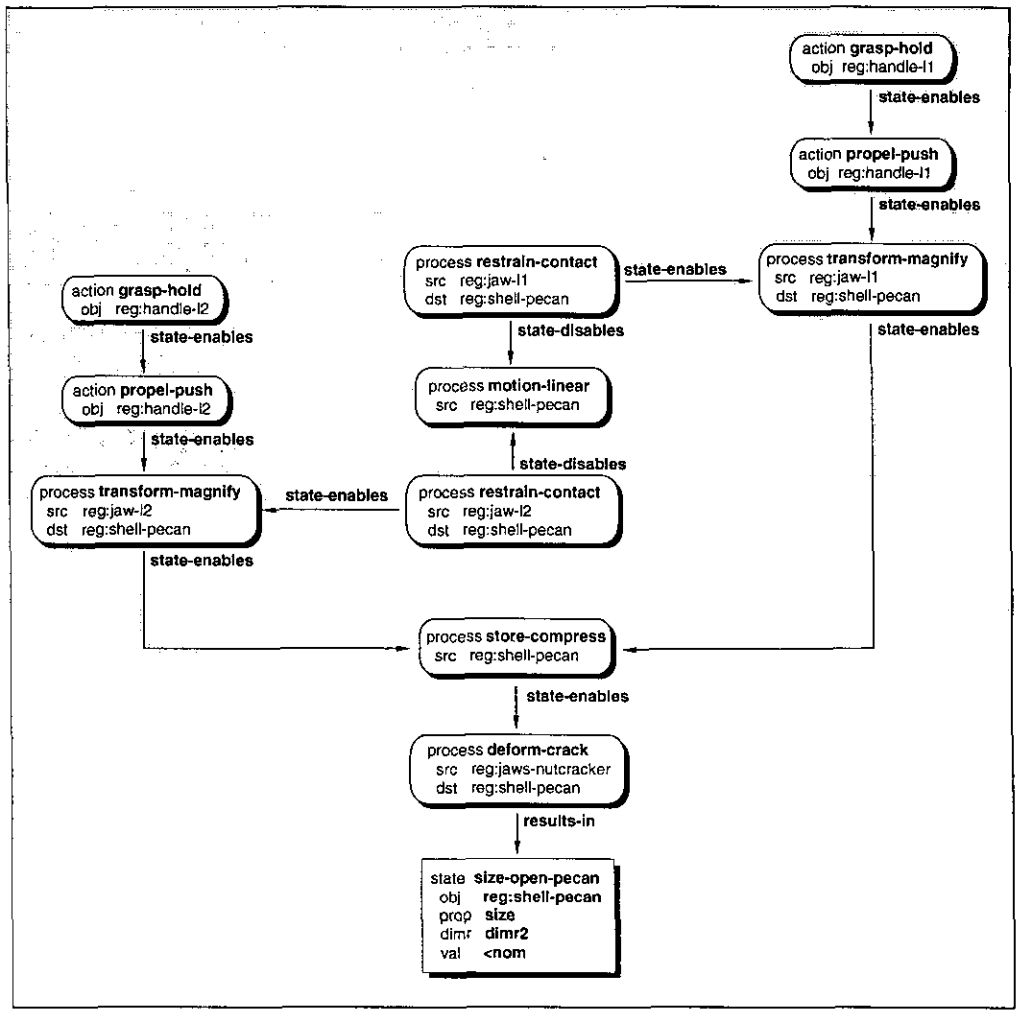

FIG. 8. Process sequencing in FONM. Oblongs depict dynamic structures (actions and processes), rectangles depict states.

will also instantiate a type 2 lever's `appl` roles, the `appl` and `react` regions of `pivot` will instantiate a type 2 lever's `pivot` role, and the type 2 lever's `react` roles will be instantiated by the new object. Arcs between the primitives represent flow of control for the device which, for most mechanical devices, is force.

Device Pragmatics

Device use is represented with a structure called a *device-use plan* and with a set of goals (behavioral delta goals and mechanical instrumental goals) that are specific to device application. A *behavioral delta goal* is a subgoal associated with bringing about an intermediate physical state in the service of a plan. There is a behavioral delta goal for each behavioral property:

- D-MOTN: Produce position changes, generally motivated when D-PROX is blocked
- D-REST: Change physical restraints on an object, generally motivated when D-CONT is blocked
- D-FORC: Produce applied forces on an object, generally motivated when PROPEL(BP) is insufficient
- D-STOR: Change object internal forces
- D-SIZE: Change physical dimensions of an object.

Behavioral goals are motivated when plan subgoals are blocked. For example, suppose you want to open a bottle of peanut butter to make a sandwich, but the lid is on too tight. The normal plan of twisting off the cap is blocked, so some other means of removing the restraint on the lid must be found to recover the plan. This new subgoal is represented with a D-REST goal, and whatever method is selected should result in the required change in restraint on the lid. A similar example is illustrated in Fig. 15, where an actor is trying to put some silver polish onto a rag so as to polish some candlesticks.

The overall goal of preserving the beauty of the candle-

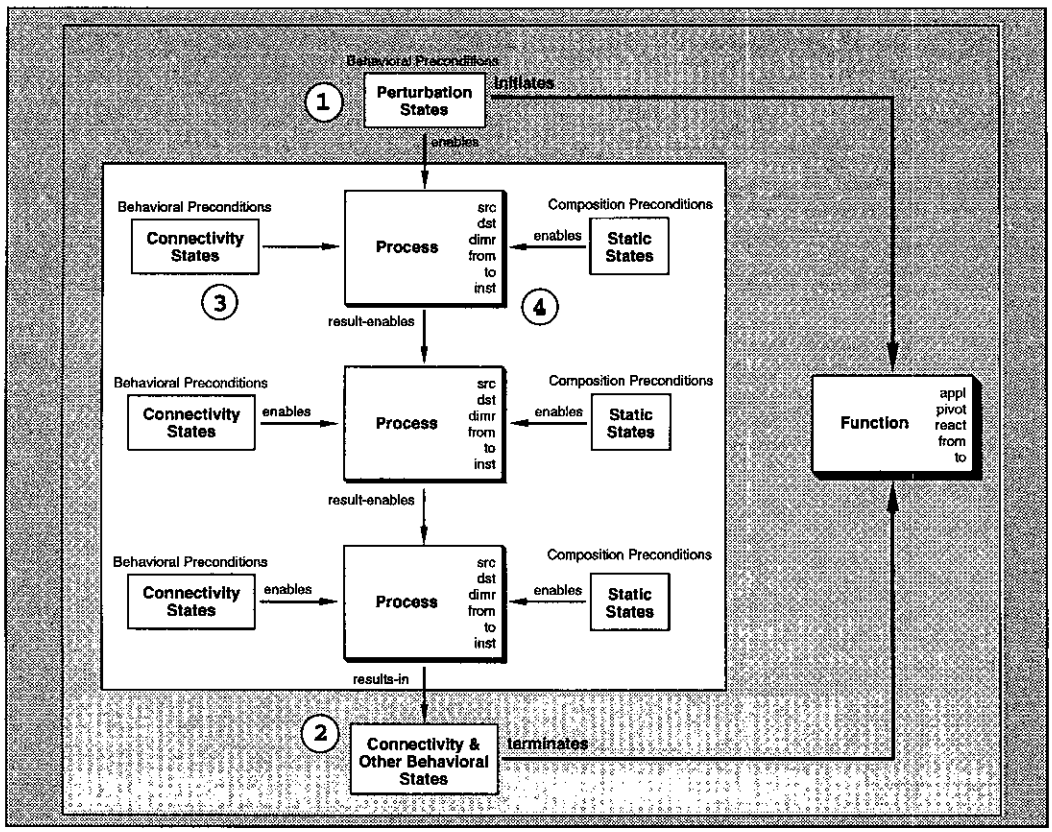

FIG. 9. Schema for representing device function in FONM.

sticks may be indexed to any number of plans, one of which may be to polish the object, as noted at **(1).** The related device-use plan has as a precondition that the actor gain physical control over all the necessary items, one of which is the silver polish itself (`d-cont` stands for delta-control, **4**). Two environmental states, the closed silver-polish can (**3**) and the silver polish being inside the can **(2),** disable efforts to gain control of the silver polish and motivate a D-REST goal to open the container (at **5**). One plan which can realize the goal is to pry the can open **(6),** which results in the open container (at **7**). In this manner, the actual states that are used to represent device statics in a particular context can be directly related to the problem-solving process.

The `pry-open-container` plan illustrated at **(6)** in Fig. 15 requires knowledge of the whereabouts and use of an object capable of prying. *Mechanical instrumental goals* identify the need for a device with a particular functionality, similar to the find(object) subgoal suggested by [10], but modified so that the goal is associated with what the object does rather than just its name. That functionality is instrumental to enabling some plan, such as the capacity to create mechanical advantage. There are five mechanical instrumental goals used in FONM:

- I-REACH: Extend control beyond a bodypart or component limit
- I-MECH: Gain purchase through mechanical force advantage
- I-CONST: Gain mechanical control of an object's range of movement
- I-RESIST: Produce passive force that requires mechanical assistance
- I-WEDGE: Produce changes in object position through mechanical position advantage.

The resolution of the candlesticks problem is further fleshed out in Fig. 16, starting from the `d-rest-open-container` behavioral goal **(5)** and `pry-open-container` plan **(6)** in Fig. 15 (here **1** and **2,** respectively). The mechanical instrumental goal, `i-mech-pry-device` **(3),** identifies that aspect of the problem that can be resolved mechanically and is satisfied by any device capable of producing mechanical advantage.

The function (at **4**) instantiates the desired function. This function can be searched for in memory and the associated devices can be looked at to see if they satisfy the other requirements of the current scenario. The result of this type of representation is a function, such as PRY-APART

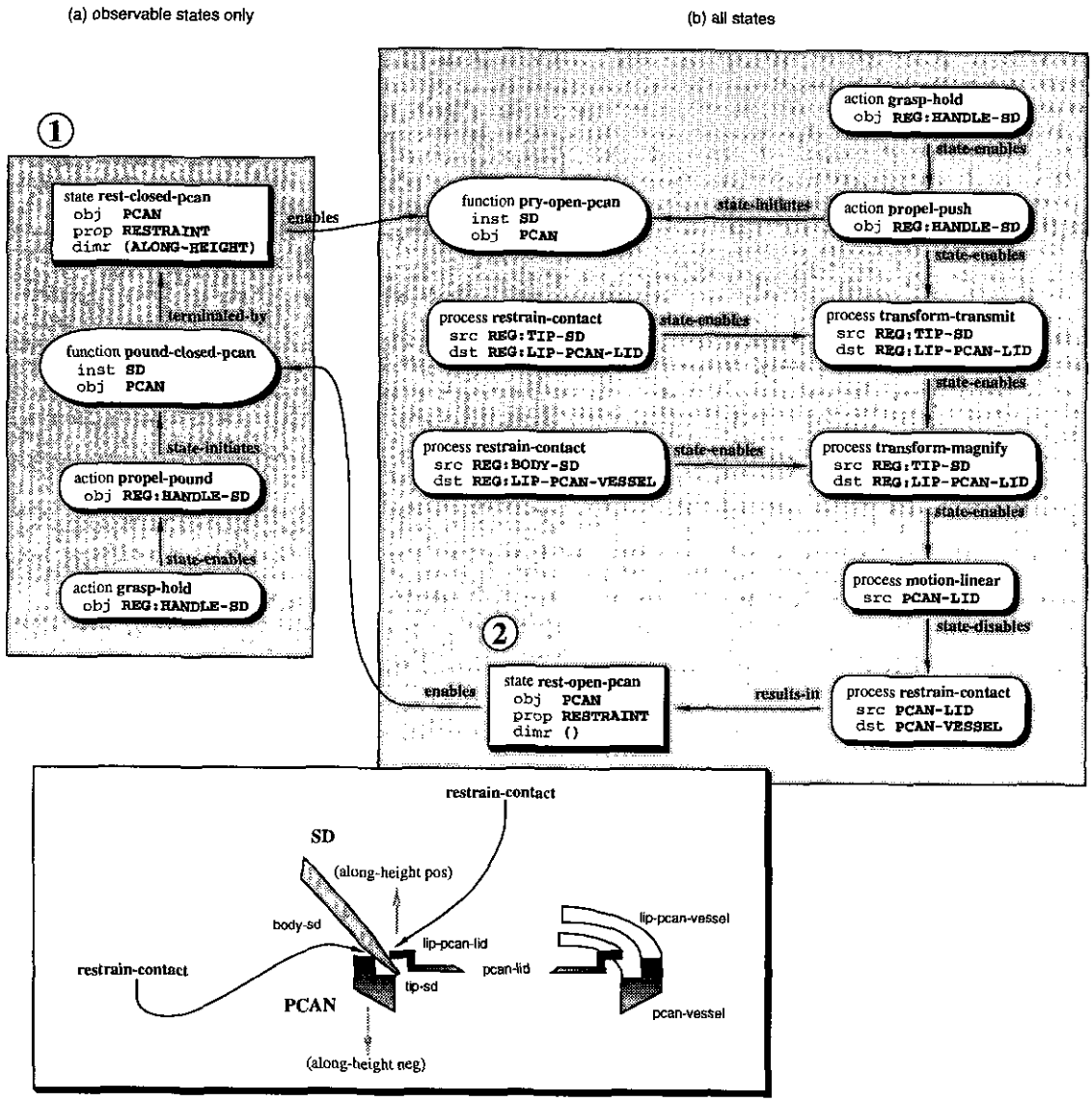

FIG. 10. Bounding function states and function granularity for screwdriver pounding and prying functions. Ovals depict functions.

shown in Fig. 17 below, which is a combination of a Lever machine primitive device and the processes which its invocation enables.

In this function, which is directly invoked in a plan such as `pry-open-container`, the `i-mech-pry-device` goal is satisfied by an instantiated MP-LEVER function. This results in an enabled rotation of the lever-object, a linear motion of the can lid, and, other physical constraints (namely the magnitude of magnified force with respect to the friction force) being satisfied, a removal of the lid restraint (identified in bold, at **3**, in Fig. 17).

Plans which use mechanical devices have traditionally considered only the human actions associated with the device rather than the device functional, behavioral, and structural requirements that must be met along the way, mostly because such representations did not exist at the time. The use(object) plan proposed in [10] combines the function and behavior of devices into an instrumental-preparation (I-PREP) subgoal which is generally supposed to be satisfied with what the authors called an instrumental script. In FONM these are made explicit by the representation of behavior and function, and the use (object) plan has been modified to support multiple objects. The resulting device-use plan can be associated with the behavioral delta subgoals, instrumental mechanical subgoals, and machine primitives associated with the plan and problem-solving context. Eight such plans have thus far been identified and are shown in Table 1.

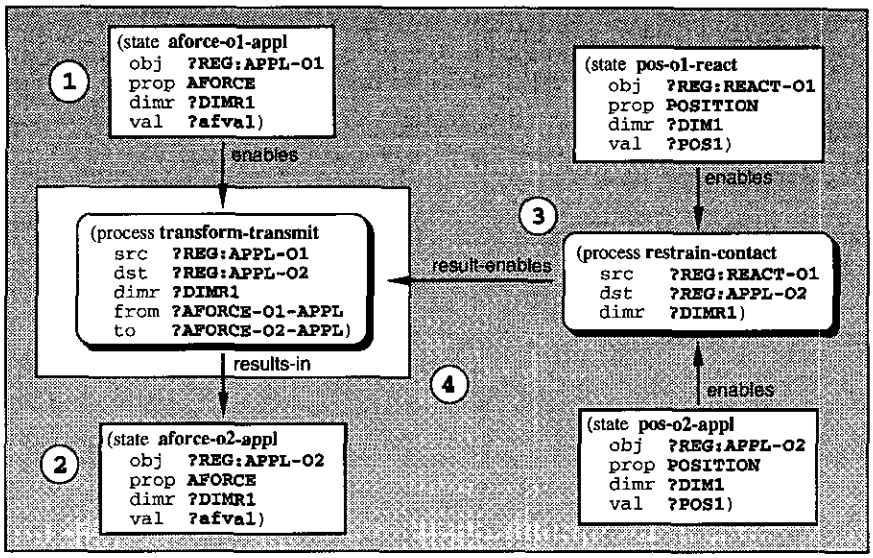

FIG. 11. Representation of linkage function using behavioral primitives and object states.

For example, in the *Hanger* scenario, the problem solver must locate a device which extends reach but also meets the other physical constraints set up by the context. The associated MCAD plan is instrumentally enabled by a device that is capable of being deformed and transmitting force. A typical device-use plan, and one resolution of the associated subgoals, is illustrated for the nutcracker in Fig. 18.

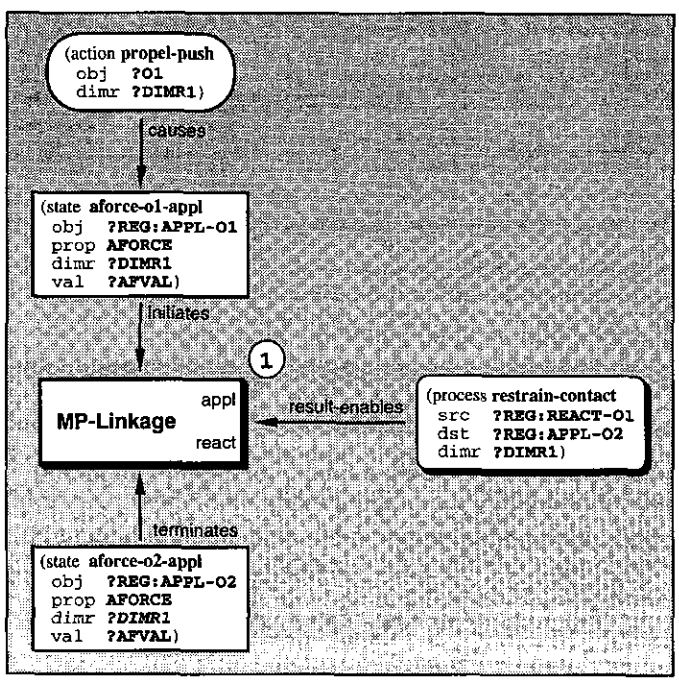

FIG. 12. MP block diagram for MP-Linkage.

In this figure, the find(nutcracker) subgoal is resolved into three instrumental mechanical goals: (1) a device capable of mechanical advantage must be located, (2) the device must be capable of containment, and (3) the device must support passive return. The preparation subgoal is resolved into a delta-restraint (placement) subgoal. These resolved subgoals are further resolved in Fig. 19. In this figure, the subgoals are resolved into instantiated machine primitives (functions, by the regions involved) for two levers, a container, and a spring, as well as another (INSERT) plan.

The representation of plans and goals for device use is only a component in the process of problem solving. Just as important is the comparison of devices at the static, dynamic, and pragmatic abstraction levels. It is often the case that two objects that appear similar are not suitable for the same tasks, and it is also common that two objects that do not appear similar can, indeed, be used in similar contexts. These ad hoc truisms suggest that comparisons between devices not be limited to a single structure or aspect of the device. In FONM, devices can be compared at both the functional (i.e., static and dynamic) level and the intentional (i.e., pragmatic) level. Devices which satisfy the same structural definitions are termed *property equivalent.* Devices that are property equivalent and instantiate the same machine primitives are termed *MP-equivalent.* Devices which satisfy the same contextual constraints are termed *contextually appropriate.* Finally, devices that are both MP-equivalent and contextually appropriate are termed *functionally equivalent.*

ATTRIBUTES AND FONM

Mechanical improvisation scenarios are an appropriate representation domain for FONM because they require

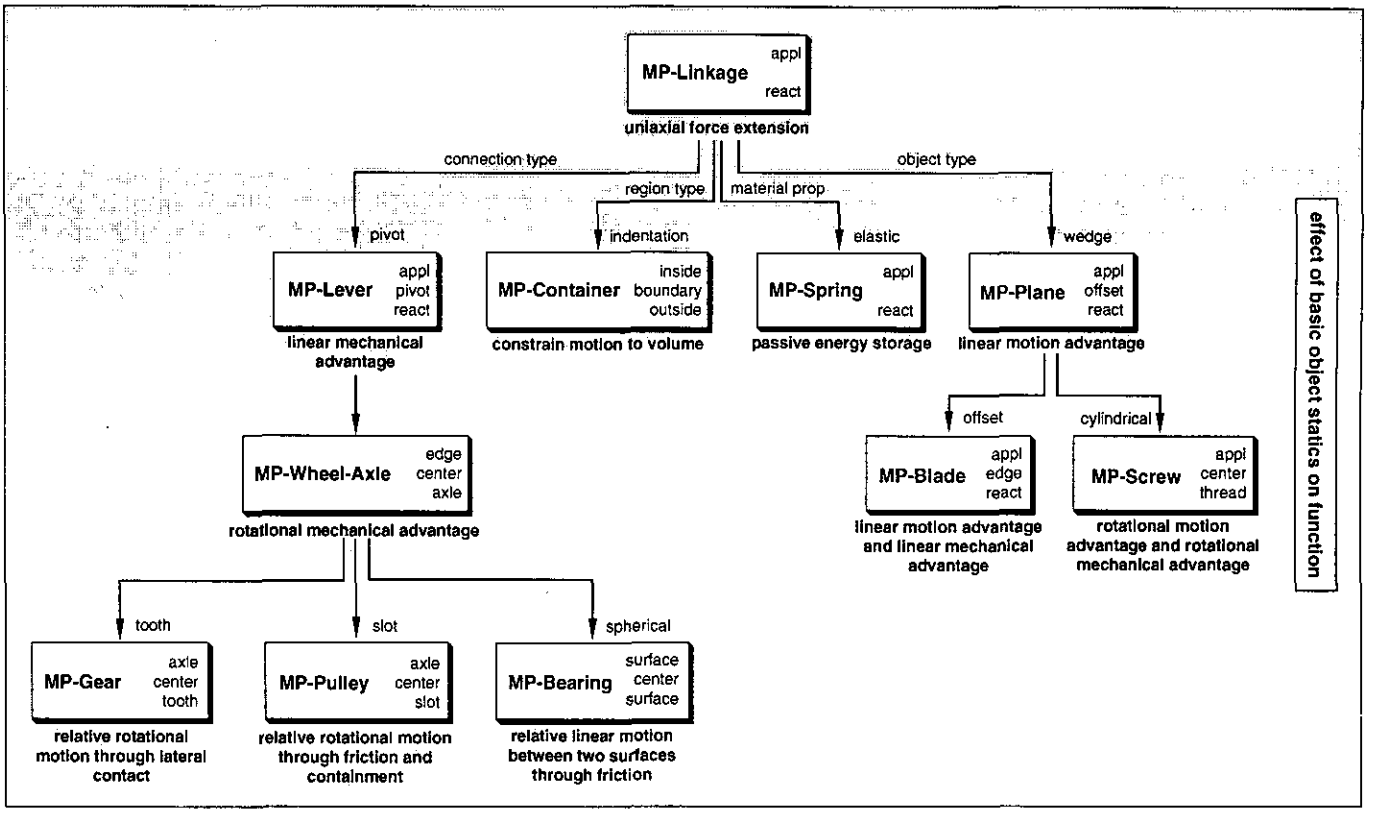

FIG. 13. Machine primitive hierarchy.

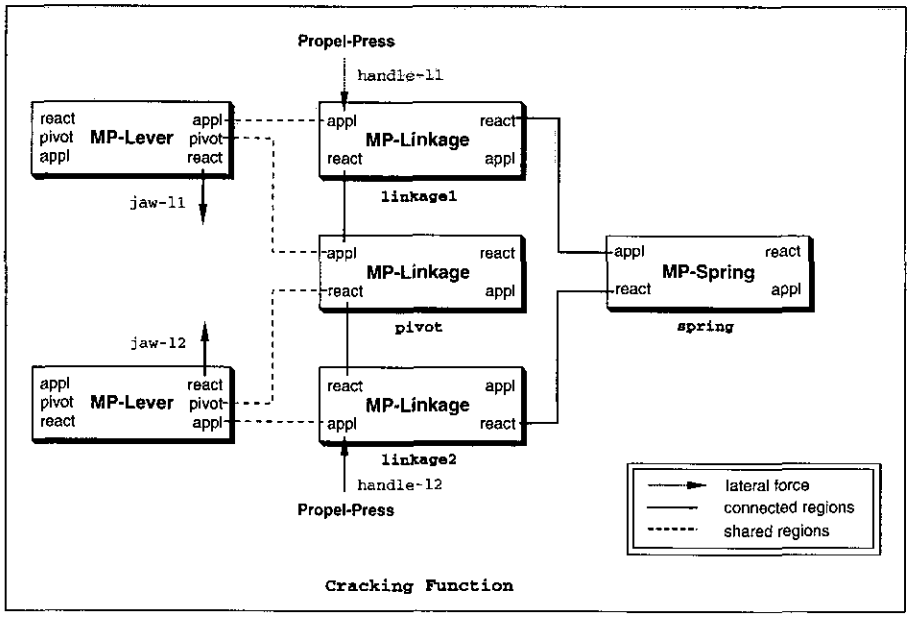

FIG. 14. Combination of machine primitives to represent nutcracker function.

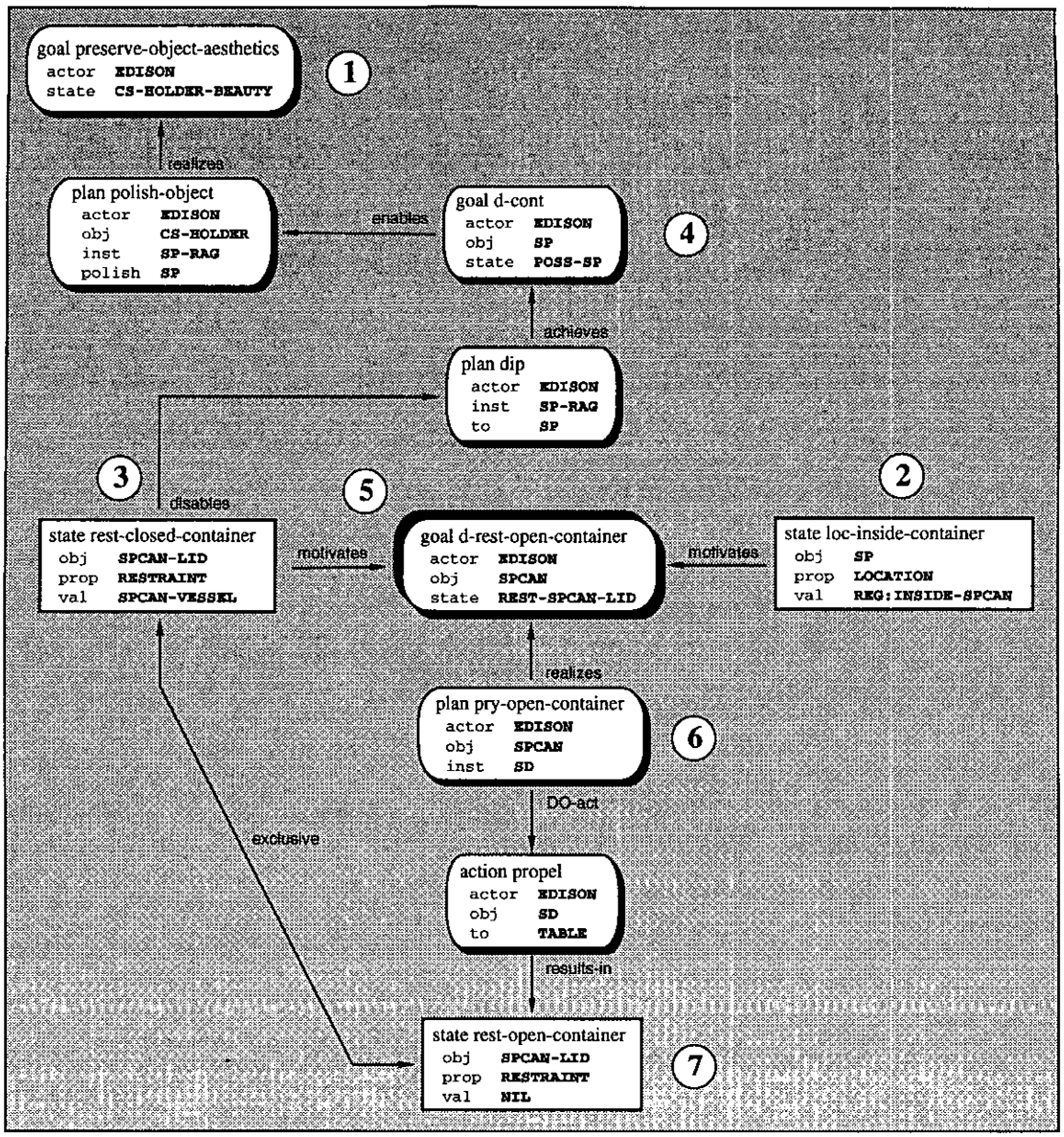

FIG. 15. Behavioral delta goals.

the problem solver to recognize in a device some functional capacity it has not been used for previously, or was not designed for. Before this can be done, the problem solver must first be able to recognize a blocked goal and an explanation for the block that is associated with the needed/available devices. Second, the functional relationship between the device and the disabled plan must be recognized. Third, in order to recover from the blocked subgoal, the plan, or device, or both may have to be adapted to the scenario. Finally, to perform this adaptation, different devices used in similar scenarios, or the same device used in different scenarios, may need to be recognized, interpreted, and compared to the current scenario.

The FONM device representation provides an integration between the static, dynamic, and pragmatic abstraction levels, so that a problem solver can access that aspect of a device which is appropriate for a specific level of reasoning. This assumes that the plans and devices are already modeled in memory. During creative problem solving, there is no mechanism for obtaining information about device function that does not already exist in memory. Nevertheless, human problem solvers routinely extract functional knowledge from objects that they have no prior experience within a particular context. Property attributes fill this gap to some extent.

Returning to the *Hanger* scenario, suppose that we seek

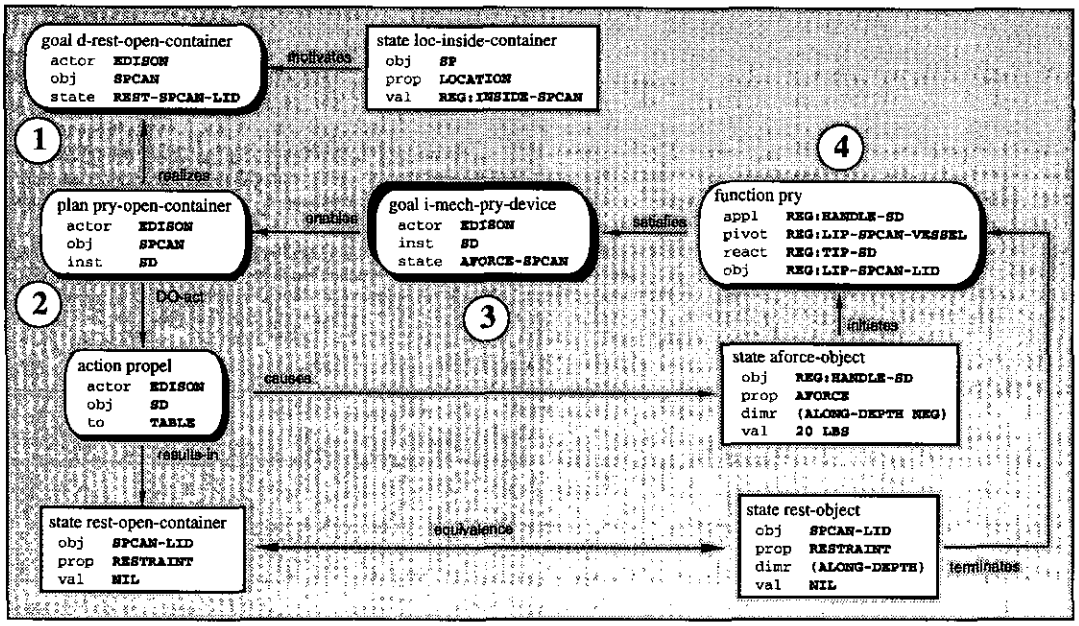

FIG. 16. Mechanical instrumental goals in problem solving.

an object that can be used in the same capacity as a hanger. Such an object would be graspable, bendable, strong, narrow, and long. None of these is an object property, but all are *functional properties*. When we find such an object we know it is successful because it works, not necessarily because we know its composition. We remember that it worked, whether or not we know why. In our memory of the experience, the object now has attributed to it whatever property associated with the hanger's functionality that is necessary in getting access to the specific vehicle. When next we find ourselves in a situtation calling for a hanger-like device, we should now be reminded of a hanger *and* this new object, because they have both been successfully applied in the same context and in service of the same type of goal. If we had experience trying objects which had not worked, we would expect to remember them as well, so that we would not repeat those failed experiments over and over again.

In FONM, the attribute is a representation construct that relates device function to the context in which it is applied without specifying the behavioral properties that causally enable the function. Thus the attribute is causally related to both the real object property and to the device function, as depicted in Fig. 20.

The importance of having a structure that is causally related to function from the intentional side of the representation is based on the fact that device function, behavior, and structure are context-free. They are not affected by a human problem solver or the location in which they are applied: if a perturbation is applied to an object it behaves in a certain way. On the other hand, plans and goals are context dependent. Thus the attribute is a context-dependent causal enablement for device function. The key term is context dependent. If an object has an attribute, then that attribute can reliably be applied as long as the original context is reinstated. However, when human problem solvers are solving problems, they often ignore the context and apply a plan if their goal has a high enough priority. Hence, a device can be applied in situations in which it has never been applied and may or may not be applicable.

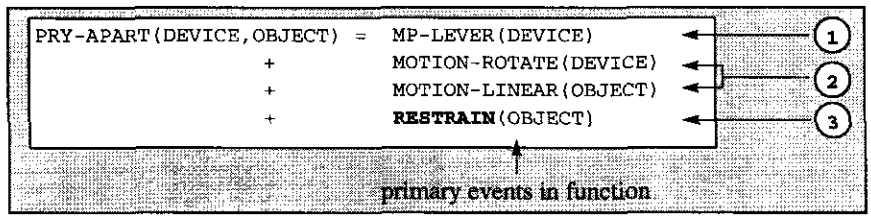

FIG. 17. The hybrid function PRY-APART.

TABLE 1

M-Plans

Plan type	BD goals	MI goals	Machine primitive
MCAD—make contact at distance	D-REST	I-REACH	MP-LINKAGE
TFD—trade-off force and distance	D-FORC	I-MECH	MP-LEVER
CFD—change force and distance	D-FORC	I-MECH	MP-LEVER
CROM—control range of motion	D-REST	I-CONTN	MP-CONTAINER
RAF—resist a force	D-STOR	I-RESIST	MP-SPRING
TMD—trade-off motion and distance	D-MOTN	I-WEDGE	MP-PLANE
SOP—separate object pieces	D-SIZE	I-WEDGE	MP-BLADE
SO—shape object	D-SIZE	I-WEDGE	MP-PLANE

Attribute Representation

An attribute is a static knowledge structure with five roles: obj, situ, prop, ref, and val. The *situ* role refers to the situational context in which the attribute was defined. The *prop* role refers to the property of comparison, when known. The *ref* role provides the state reference for the comparison. In Fig. 21 the attribute ATT-STRONG-HANGER is used to represent the strength of the body of a hanger, but the reference state is the friction force between the door lock knob and the door. We say that the hanger is "strong" as long as the breaking strength of its body is greater than the friction force, because, in the *Hanger* situation, that is the force needed to unlock the door.

Attributes and Object Properties

The ability to make correspondences between attributes and property values is important because of the different inferences that can be made at the functional and behavioral levels, respectively. The correspondence is based on the notion that each attribute describes a *range* in a property's quantity space. The correspondence is weak (i.e., one to many) but real. Many attributes are associated with object function through a specific property, such as *heavy* to weight, or *long* to length. Attributes can also be represented as combinations of properties or other attributes. The attribute *metallic*, for example, is a combination of the attributes *shiny*, *smooth*, *cold*, and *hard*.

There are two ways that attributes are referenced to property values in quantity space:

- To a known reference point or value (e.g., silver-polish can-to-lid friction force).
- To a boundary value (e.g., the full-open position of a water faucet).

The reference point defines the context for which an attribute is directly applicable and is always found in the situational context. For instance, in *Hanger* the device selected must be *narrow* enough to fit in the gap between the car frame and the window, it must be *long* enough to reach the door lock knob, it must be *malleable/bendable* enough to shape it so that it can reach the door lock knob, it must be *strong* enough to transmit enough force to overcome the friction between the door lock knob and the door, and it must be *graspable* so that the person trying to gain access to the vehicle can manipulate it. In each of these attributes, the reference is identified as a physical constraint imposed by the environment (window gap, distance to door lock knob, maximum bend radius, friction force, and size, respectively).

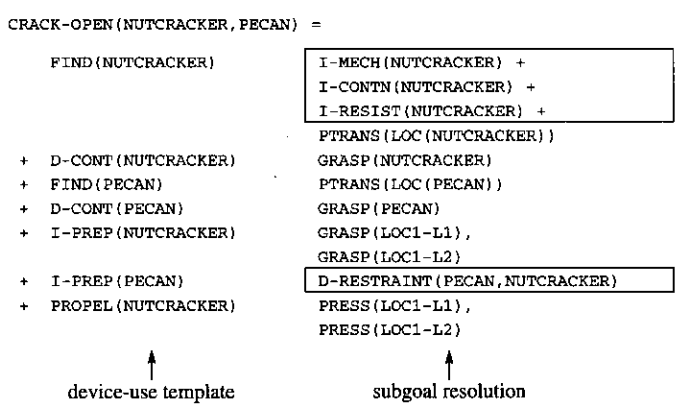

FIG. 18. Device-use plan for a nutcracker in its standard nutcracking application.

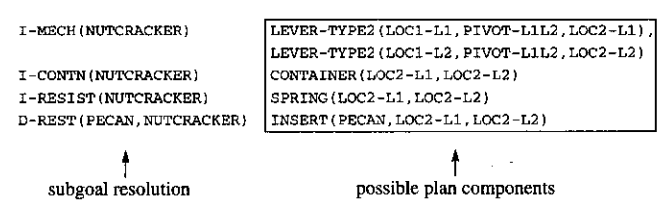

FIG. 19. Continued subgoal refinement for nutcracker crack-open device-use plan.

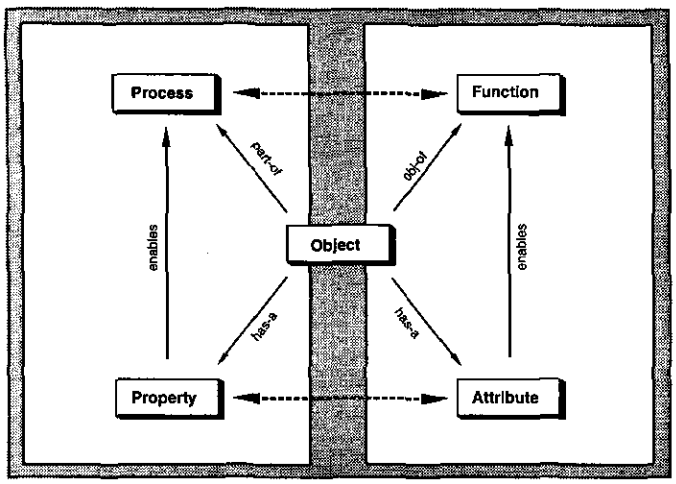

FIG. 20. Attributes and their causal relationships to other FONM knowledge structures.

Attributes and Object Function

An attribute sidesteps the direct relationships between function, behavior, and structure, replacing them with the results of practical experimentation. Certainly if an object has been successfully applied in a functional context it can be again. The attribute is directly causal to function as long as the context is reinstated. When a problem-solving scenario is recognized, attributes can be used in two ways. First, attributes can be used to recognize the devices in the environment, because they provide generalized definitions of object properties, and they are generally related to specific regions on the object. Second, once recognized, the devices which are available can be looked at to see if the attributes they have from their own uses in different experiences match the constraints of the current scenario. As experience with using an object is broadened (i.e., different contexts), the difference between the attribute and the physical property is reduced. Eventually the problem-solver's experiential knowledge of the object could have a near 1 : 1 correspondence with behavioral properties.

Object attributes are also associated with functional locations on an object and can affect the interpretation of device function. For example, a *sharp* edge represents the quality of a region. The very mention of its sharpness provides evidence for recognizing functions of that region associated with the quality. In a similar way, any aspect of the device static representation might be associated with an attribute of a device.

```
(attribute ATT-STRONG-HANGER
   obj     REG:BODY-HANGER
   situ    HANGER
   prop    BREAKING-STRENGTH
   ref     (state FFORCE-CARLOCKKNOB
              obj    EDGE-CARLOCKKNOB
              prop   FRICTION-FORCE)
   val     >)
```

FIG. 21. Attribute representation for a strong hanger.

Attributes and Object Use

Planning is a goal-directed activity, and plans are often selected based on the strength, or priority, of a goal. For example, if a person's life is in danger, s/he might attempt otherwise intractable plans simply because the result of failure cannot be much worse than the given situation. Attributes affect planning in two ways. First, an object's attributes direct planning choices in context by constraining applicable functions. At the intentional level, attributes describe functional capabilities of an object learned through experience and are specific to particular objects in particular contexts. Knowing a device has an attribute enables high-level inferences about the device's functional capabilities if the context is reinstated. For example, knowing that a hanger was successful in unlocking a door, one might have concluded that the hanger is a strong object (w.r.t. the door-lock friction). The *strong* attribute of the hanger is only valid for this situation. Other situations requiring *strong* objects, however, might remind the problem solver of the hanger. Second, a plan may be selected based on the importance of a goal, or the relationship between goals, and the original context of an attribute may be ignored. This can lead to failure, because the selection is effectively an experiment. It can also lead to innovation for the same reason.

THE HANGER SITUATION

The representation for (some of the) functional and behavioral inference paths in *Hanger* are fleshed out in Fig. 22. The behavioral representation shown in the figure represents the process interactions supporting the UNLOCK-OBJECT function with the hanger instantiated as the JIMMY-OBJECT. The representation is shown instantiated with the hanger after retrieval from memory and combination into the UNLOCK-OBJECT function. The attribute/property-value relationship is shown as it affects the functional/behavioral description under the heading Attribute-Property Mapping. The *fit* requirement affects UNLOCK-OBJECT in two dimensions, so the constraint on motion is made in two dimensions.

RELATED WORK AND SUMMARY

FONM is a general model that enables the representation of complex mechanical devices across the spectrum from structural characteristics to functional equivalence and adaptation in problem solving. The model provides taxonomies of generalized, discrete structures for the structural-behavioral, behavioral-functional, and functional-contextual abstraction levels. Thus the knowledge at differ-

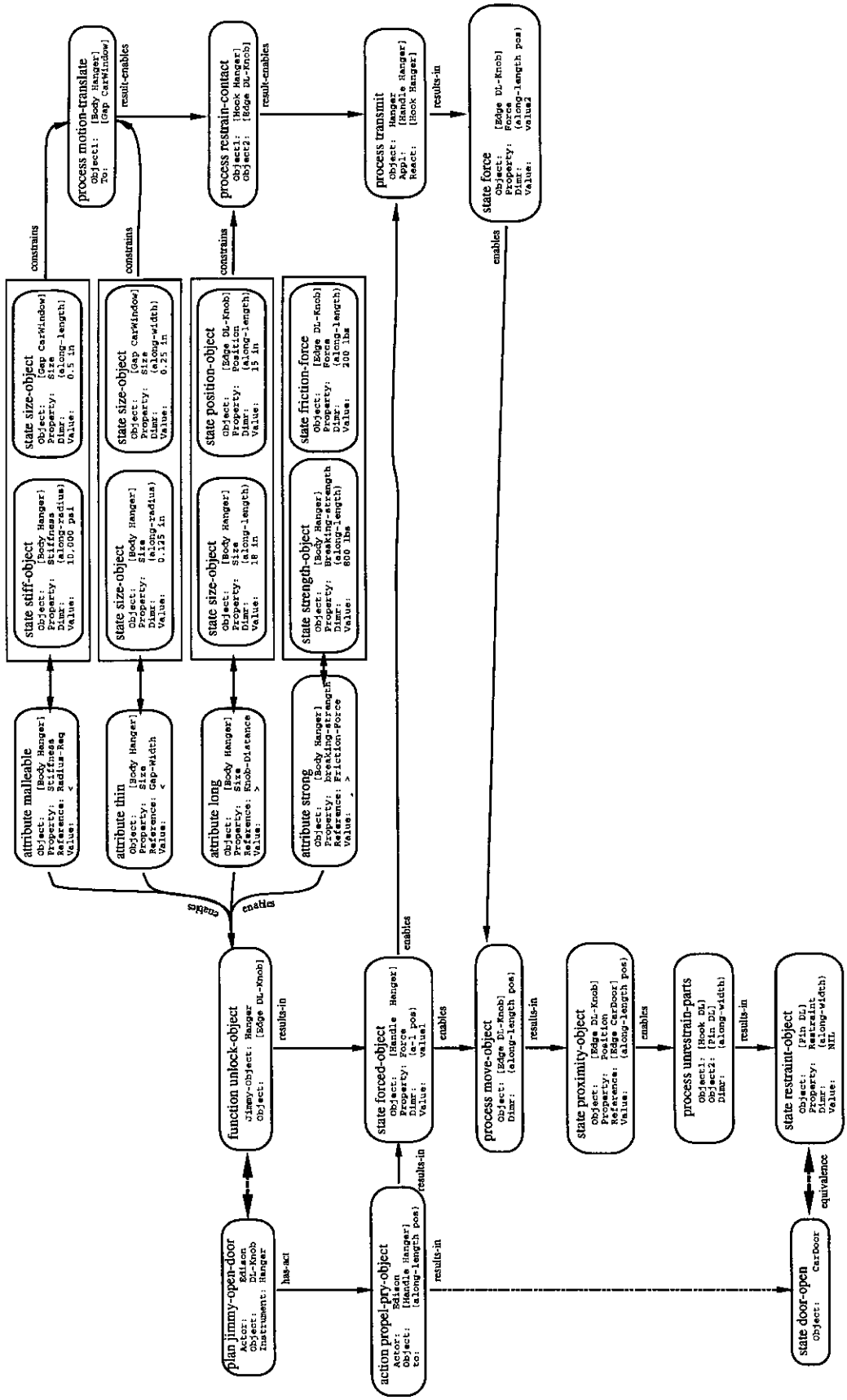

FIG. 22. Attribute-Property Mapping: Attribute-use-function representation applied to the *Hanger* scenario.

ent levels of abstraction remains independent but causally related, supporting their inclusion and instantiation at whatever level they become required. Some FONM constructs, notably regions, machine primitives, device-use plans, and property attributes, have already been employed in object recognition research. Regions provide function-based locations on an object that are related to shape. The GRUFF project has utilized the notion of regions to recognize articulated objects such as scissors, and even to recognize a pair of scissors that cannot work [4]. Machine primitives provide a relationship between object regions and function. A number of projects are using machine primitives (or similar) to assist in recognizing objects based on their interactions with other objects [2, 6]. Device goals and plans provide a description of how and why devices are used. Some researchers have been using the notions of device-use plans to aid in recognizing whether the device can be recognized [1, 2, 6]. Finally, property attributes provide a means of relating function to what an object has been used to accomplish or might be used to accomplish. Various kinds of attributes (under different guises) have been extensively applied in recognizing objects [1, 4, 6, 11]. To date, most function-based object recognition models make use of geometric attributes, such as graspable, sittable, and fit, though the potential for defining and using attributes in this capacity has not been fully explored.

Researchers in object vision generally point out that the many-to-one mapping of attributes to function makes recognition difficult. However, in problem-solving domains this mapping is a blessing, because it provides a mechanism for explaining stupidity and innovation. One aspect of the FONM model that has not been applied, and might help in ameliorating this conflict, is the notion of functional equivalence as a combination of MP-Equivalence and Contextual Appropriateness. MP-Equivalence is a measure of equivalence based on whether a device can instantiate a machine primitive and includes the context-free aspects of the representation. Contextual Appropriateness is a measure of a device's attributes and how they relate the device to the current environment. Whereas a simple application of attributes to the recognition process might lead to too many possibilities, the application of this definition of functional equivalence might not.

To date, FONM has been used primarily to model the function of devices and the use of devices based on their functional representations. Object regions and attributes play fundamental roles in supporting reasoning about devices in these contexts. Vision research on object recognition using function has shown that the definition of shape-based property attributes can be useful for recognizing objects. Although computational models exist for demonstrating some of the lower representation levels of FONM, an integrated model which supports the full representation breadth is yet to be implemented by the author.

REFERENCES

1. L. Bogoni and R. Bajcsy, Functionality investigation using a discrete event system approach, *J. Rob. Autonom. Syst.* **13,** 1994, 173–196.
2. L. Birnbaum, M. Brand, and P. Cooper, Using causal scene analysis to direct focus of attention, in *Proceedings of the IEEE Workshop on Qualitative Vision, New York, June 1993,* pp. 23–32.
3. K. Forbus, Qualitative process theory, *Artif. Intell.* **24,** 1984, 85–168.
4. K. Green, D. Eggert, L. Stark, and K. Bowyer, Generic recognition of articulated objects by reasoning about functionality, in *Proceedings of the IAPR International Conference on Pattern Recognition, Jerusalem, Israel, Oct.* 1994, pp. 847–849.
5. A. K. Goel, Integration of case-based reasoning and model-based reasoning for adaptive design problem solving, Ph.D. dissertation, Computer Science Department, Ohio State University, 1989.
6. K. Kise, H. Hattori, T. Kitahashi, and K. Fukunaga, Representing and recognizing simple hand-tools based on their functions, in *Asian Conference on Computer Vision, Osaka, Japan, Nov. 1993,* pp. 656–659.
7. J. B. Hodges, Naive mechanics: A computational model for representing and reasoning about simple mechanical devices, Ph.D. dissertation, Computer Science Department, Technical Report CSD94001, University of California at Los Angeles, 1993.
8. J. B. Hodges, Naive mechanics, *IEEE Expert* **N**(M), Feb. 1992.
9. J. B. Hodges, Device representation for modeling improvisation in mechanical use situations, in *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society, Ann Arbor, MI, Aug. 1989.*
10. R. Schank and R. Abelson, *Scripts, Plans, Goals, and Understanding,* Lawrence-Erlbaum, Hillsdale, NJ, 1977.
11. L. Stark and K. Bowyer, Achieving generalized object recognition through reasoning about association of function to structure, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(10), 1991, 1097–1104.