NAIVE MECHANICS:

A Computational Model of Device Use and Function in Design Improvisation

Jack Hodges
San Francisco State University

Abstract

The EDISON project is concerned with developing computational models capable of creative reasoning and problem solving with respect to mechanical devices. Two general issues are addressed by the project: (1) the breadth of and interactions between knowledge required for creative reasoning tasks, and (2) the amount of knowledge required to describe mechanical devices, their behavior, function and use. The overall approach is to represent devices with a set of discrete symbolic structures: device behavior is represented with a set of five behavioral process primitives; device function is represented with a set of eleven machine primitives; devices are represented with combinations of machine primitives; and device use is represented with plans. These device representations are used to represent problem-solving situations from different perspectives. Mechanical improvisation is supported by applying the concept of device functional equivalence for adapting devices used in one situation to different situations.

1.0 Introduction

Creative problem solving requires knowledge of object behavior, function, and use. A human problem solver calls upon experiences to match context and object applicability, experiments to see if an object will work, can recognize behavior resulting from those experiments, and can predict the function and behavior of objects previously used. A computer model capable of mechanical creativity must be able to call upon mechanical knowledge at the level (behavioral, functional, or intentional) needed to solve a problem. Moreover, the computer model must be able to use knowledge at these different abstraction levels interactively, thus this knowledge must be integrated in the model. This mechanical knowledge and reasoning typically associated with human problem solvers is called *naive mechanics*. The issues raised in building computational models for naive mechanics can be illustrated with a simple problem-solving situation requiring only the recognition that two objects are functionally equivalent for the task of cracking nuts.

Nutcracker

A man wishes to crack open some pecans for a pie he will bake for Thanksgiving. He cannot find the nutcracker he generally uses and, rather than look outside the kitchen for the nutcracker, decides to try something else. He tries a crank can opener.

Nutcracker describes a situation requiring problem-solving skills and creative thinking. The problem solver has a problem, the solution of which is bound by specific contextual constraints. We call **Nutcracker** an improvisation situation. The representational and processing requirements that situations place on a computational model are illustrated by a few questions a human problem solver might answer.

Q1: How does the Thanksgiving holiday affect the man's choices?
A1: He may be too busy preparing other things to look for the nutcracker.

Q2: What does he consider to be a suitable replacement for the nutcracker?
A2: An object which will *crack* the pecans open.

Q3: Why does he think of the crank can opener?
A3: The crank can opener and nutcracker are both kitchen utensils, and both apply *leverage* in the same way.

Q4: Why does he think that a crank can opener might work as a nutcracker?
A4: He knows that the crank can opener has two *handles pivoted* together at one end, like the nutcracker. He knows that the crank can opener has been used to initiate a *cut* in soup cans, so it is *strong* enough. He knows that the handles can be spread wide enough to *restrain* the pecan, and that the pecan can be *held* in place with his other hand during cracking.

Questions **Q1-Q4** illustrate three reasoning levels which the man in **Nutcracker** uses to resolve his problem: behavioral, functional, and intentional.

Behavioral Reasoning: Low-level object state to state (temporal) change is called *object behavior*. Answer **A4** illustrates some knowledge of object behavior. The man knows something about pivoted objects, restraint, and cutting. Each of these concepts describes object behavioral dependencies. When he chooses the crank can opener as an alternative device, he does not know whether it will work. When he tries the crank can opener, he remembers its behavioral response with respect to the current goal and plan context. The problem solving session thus becomes a new situation in memory which he might later recall.

Functional Reasoning: When faced with the lack of an appropriate device for nutcracking, the man decides to adapt the plan by using an alternative device. Answer **A2** shows that he associates cracking with opening, and that he has some knowledge of cracking and how devices might be used to cause the desired state. One way to select a device is based on whether it has been used in the same capacity in other situations. Another is to select a device which has similar functional capability. Answer **A3** shows that the man can recognize devices by how they apply leverage. In seeking a similar application of leverage, his choice of devices is reduced. Answer **A4** illustrates knowledge of how the nutcracker and crank can opener functions are initiated and what states are caused as a result.

Intentional Reasoning: Answer **A1** illustrates how the man's goals and planning choices are affected by the situation: the need to interpret the man's plan within the context of a major holiday, and the need to associate mechanical objects with the kitchen setting. By recognizing contextual constraints on actor goals and plans, the effective number of planning choices is reduced, making the computational problem more tractable. Answer **A4** shows that the man makes his choice of devices partially on how they match the context of the cracking-nuts-with-nutcracker plan: the need for handles, the size of opening to match the size of nut, the strength of the device, and holding the nut while using the device.

Behavioral, functional, and intentional reasoning require the human problem solver to access different kinds of knowledge. In order to construct a computational model with similar capabilities, a representation approach is required which describes the breadth and amount of knowledge needed in problem solving. This article presents an approach for representing mechanical devices and mechanical problem-solving situations. The **Nutcracker** example is used to show how the approach supports the representation and reasoning requirements of naive improvisation.

2.0 EDISON Approach to Device Representation

Improvising solutions to mechanical design problems requires knowledge about how devices function. For humans, much of this knowledge is gained from experience using devices in various ways. The goal of the EDISON project is to represent and manipulate problem-solving experiences, called *situations*, in which mechanical devices are used. This goal is accomplished by representing device behavior, function, and use with a discrete number of knowledge primitives. Primitives can be combined to describe complex behaviors and functions. Devices so represented can be applied to many tasks, and devices which share primitives can be applied to similar tasks. The primitives are causally related so that knowledge about device use enables access to knowledge about device function and behavior. EDISON device representations can then be used in reasoning tasks requiring description, prediction, inference, and explanation.

2.1 Knowledge Primitives in Naive Mechanics

The EDISON model uses conceptual primitives to represent the knowledge and reasoning associated with mechanical objects. Because primitives are used to represent a large number of related concepts with simple, basic patterns, their use has been effective for constructing computational reasoning models. For example, in Conceptual Dependency (CD), [Schank, 1977], actions are knowledge primitives describing causal interactions between intentional agents. Actions can be combined to describe complex intentional behavior, as represented with scripts, plans, and goals.

In EDISON, the causal interactions between a problem solver and an object are represented using *causal dependency*. Causal dependency extends the notion of CD to interactions not involving intentional agents. Grasping, moving, and pressing are examples of CD actions associated with a plan to open pecans. They produce expectations for nutcracker movement and the open(pecan) state. To describe nutcracker function, EDISON must be able to recognize that the nutcracker has different functions, depending on the context of its use, and know how to apply them. EDISON can then recognize that the actor's actions initiate the nutcracker function crack-open(pecan). The causal relationships that occur within the nutcracker describe its behavior and result in the observable states associated with successful plan execution.

2.2 Representation Layers

In EDISON, devices are represented in three layers, each describing a different degree of causal granularity: (1) device physical characteristics and composition, (2) device behavior and function, and (3) device use. Each layer describes different inferences that can be made about devices during problem solving.

Devices are comprised of one or more individual objects connected together. An object is described by its physical characteristics, and configurations of objects are described by their relational characteristics.

Device behavior is characterized by a set of five *behavioral process primitives* (BPPs), which represent interactions between individual objects and produce changes in state. Behavioral primitives are distinguished by the type of state they produce. BPPs are specialized according to the physical and relational characteristics of participating objects. Complex mechanical behavior can be represented as causal sequences of BPPs. Device function is associated with BPP sequences which are initiated and terminated by observable states.

Device function is characterized by a set of eleven *machine primitives* (MPs). Each MP is associated with a specific object type and function (behavioral sequence). Complex mechanical device functions can be described as a MP combinations. Each MP can be perturbed in different ways, and each MP comprising a device can be considered independently of the other MPs in the device representation. Thus each device can have many functions, each represented with MPs instantiated in different ways.

Device use is characterized as a plan, and device functions are applied as subgoals in executing the plan. Two subgoal types are used to relate device use and function: behaviorally motivated goals are used to perform low-level mechanical tasks like connecting objects, and mechanically motivated goals are used to perform higher-level tasks requiring the use of a particular machine, such as a wheel. Device use is enabled by an object's property *attributes*, which represent the comparison of property values in a specific context.

Figure 1 illustrates the dependencies between the representational constructs used to describe object behavior, function, and use in EDISON.

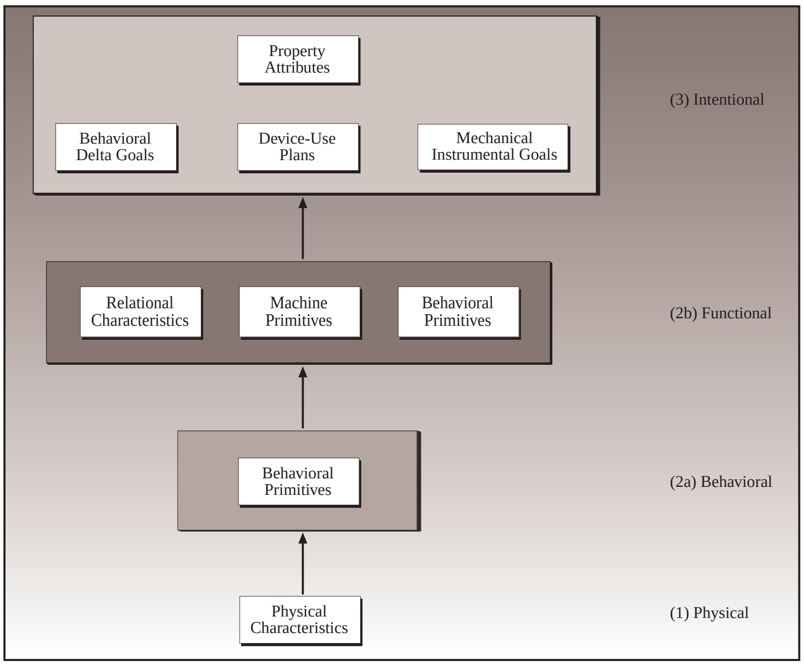


Figure 1: Device representation constructs and their dependencies, by layers.

A single object is called an *individual object*. Individual objects are represented as a collection of physical characteristics, as depicted in layer 1 in Fig. 1. Changes in an individual object's characteristics describe and constrain object behavior through behavioral primitives. Complex mechanical behavior is represented as causal sequences of behavioral primitives (layer 2a in Fig. 1). A simple mechanical object is associated with a single behavioral sequence, and the combination is called a machine primitive. Device functions are represented as machine primitives and other devices connected together (layer 2b in Fig. 1). Object use is related to object function by the properties of the objects in a particular context. Object choice in planning is constrained by the object's attributes and their relationship with the problem-solving goals.

2.3 Individual Objects

The representation of device behavior depends on the types of individual objects involved, their characteristics, and their relations to one another.

There are two classes of individual objects in EDISON: *linkages* and *stuff*. Linkages and stuff are differentiated by their material stiffness. A *linkage* is an elastic object; it is made of a material capable of transmitting force in at least one direction along at least one of its dimensional axes. That is, linkages have a non-zero property value for material stiffness in a dimension and direction.

Wires and ropes are examples of linkage objects in a single dimension/direction, whereas pipes and plates are examples of linkage objects in all dimensions and directions. *Stuff* describes objects which are not elastic in any dimension or direction, and therefore cannot be pushed or pulled on to transmit force. Liquids and gases are examples of stuff, since they require a container to effect mechanical force transmission.

2.3.1 Physical and Relational Characteristics

The behavior of individual objects is affected by their physical characteristics, and by their relational characteristics with other objects. An object's physical characteristics describe its composition and its outward appearance. In EDISON, an object has two types of physical characteristics: *property values* and *regions*. Object property values describe the object composition, and determine how the object can and will behave under various conditions. Object regions describe the object appearance and determine how the object can be combined with other objects and manipulated. In EDISON, an individual object is completely specified by its physical properties and regions.

A relational characteristic describes a spatial relationship between two objects. Object relational characteristics constrain the interactions in which objects can participate. For example, one object cannot support another unless it is placed below the other object. In EDISON, two relational characteristics are used: *orientation*, and *placement*.

Physical Properties and Physical States

Every object has a finite number of physical properties, such as size, weight and location. Each object property has a single value for any particular instance in time. In EDISON, an object's property values are represented as states, and changes in property value are reflected as changes in state. Each property directly affects how the object can behave, so changes in property value are important for describing object use and function. A state is represented with a knowledge structure having three roles: *prop*, *dimr*, and *quantity*. The *prop* role names the property being represented, the *dimr* role describes the dimension and direction of interest, and the *quantity* role describes the property value.

Object Regions

People tend to describe the appearance of individual objects in terms of those general locations having functional significance. A door handle, a table top, and a gear tooth are all general locations associated with functions of their respective objects. In EDISON, a *region* is a knowledge structure for representing locations on objects which directly constrain object behavior and function. A region has two roles, *dimr* and *value*, for describing its location on the object.

The nutcracker device depicted in Fig. 2 is comprised of four individual objects: two linkages (O:NC-LINKAGE1, O:NC-LINKAGE2), a pivot piece (O:NC-PIVOT) with which the linkages are pivoted, and a spring (O:NC-SPRING). All four components are linkage objects, since they are all capable of transmitting force.

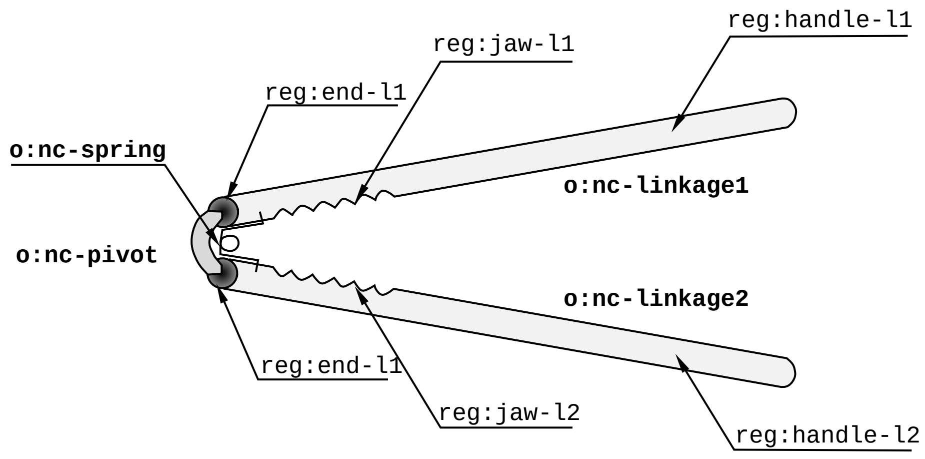


Figure 2: Four components in a nutcracker: two linkages, a pivot, and a spring. The handle and jaw of each linkage describe regions of input or output. Parts are also connected at regions. The linkage and pivot ends are points of connection and force transmission.

Each nutcracker component is described by representing those regions which directly support the component's role in any particular nutcracker function. The nutcracker is a hand-operated device, so it must have some location (e.g., REG:HANDLE-L1 on O:NC-LINKAGE1) to operate the device (i.e., grasp and press). It must also have some location where the force which cracks the nut is produced. A jaw is used to hold objects, and is intended to be the location on the nutcracker where the nut is held. Force can be transmitted at contact locations, so the jaw region REG:JAW-L1 on O:NC-LINKAGE1, wherever it is physically located, is a necessary region for nutcracking. O:NC-LINKAGE1 is connected to other objects, so the location of that connection must also be represented. This location is called REG:END-L1. Each nutcracker linkage is represented by three regions: a handle, a jaw, and an end. Because O:NC-LINKAGE1 and O:NC-LINKAGE2 are completely specified as a linkage objects with three regions, this approach simplifies the representation of an otherwise complex device.

There are two region types in EDISON: *surfaces* and *boundaries*. A surface region describes an affected area or volume, and a boundary describes the junction of surfaces. From these, all other regions can be described. There are three types of surface regions: surfaces associated with flat volumes are called *faces*; surfaces associated with containment volumes are called *indentations*; and surfaces associated with convex volumes are called *protuberances*. Faces, indentations, and protuberances are also distinguished by their size and shape. For example, a crack is represented as a long, shallow indentation, while a hole is represented as a deep indentation. An end is represented as a protuberance at an extreme location of a long object, whose length is distinctly greater than its width or depth. In the nutcracker pictured above, REG:HANDLE-L1 and REG:HANDLE-L2 are end regions. Handle regions are specialized by size, since any long object can have ends, but only ends which can fit into one's hand can be handles.

DeKleer and Brown introduced conduits to denote the transmission of force, energy, or material from one object to another, [DeKleer, 1983]. A connection between objects serves as a conduit (or channel) for force or momentum transmission. In terms of connection, conduits and regions are equivalent because regions are the locations where behavioral interactions take place.

2.4 Device Behavior

The man in **Nutcracker** needs leverage to crack the nuts open. In this situation, the result of leverage can be represented as an increase in force applied on the nut. Changes in state describe object behavior, and the dynamic interactions which produce object behavior are essential in predicting and diagnosing object function. The regions at which force transformation takes place constrain an object's capacity to engage in behavior. For example, the nutcracker O:NC-LINKAGE1 and O:NC-LINKAGE2 components illustrated in Fig. 2 are pivoted to O:NC-PIVOT at their end regions. This connection enables them to rotate independently; however, all three objects must translate as a unit. When the jaw regions are used to restrain nuts, the pivot also enables leverage by magnifying the applied force. Pivoting, rotation, translation, and magnification are examples of object behavior.

In EDISON, object behavior is described symbolically with qualitative processes. A qualitative process describes causal dependencies between objects, and produces changes in the physical states of the affected objects. Qualitative processes are represented as knowledge structures with six roles:

A **process** has:
- a **src**, a source object or region,
- a **dst**, a destination object or region,
- a **dimr**, a dimension and direction,
- a **from**, a state from which the process originates,
- a **to**, a state in which the process results, and
- a **medium**, an object or region which mediates the process.

The *src* and *dst* roles describe the locations where the interaction takes place. The *to* role states resulting from an enabled process are *local* states, because they are associated with the region at which the process takes place. The union of object local states defines a *global* state for the entire object. For example, the union of local force states on an object describes the object global force state.

The process representation can be statically illustrated by considering the action of squeezing the two nutcracker handles together in the absence of a nut. The transmission of force from each linkage to the pivot piece is represented using a process which describes force transformations, called BPP-TRANSFORM.

```
(process BPP-TRANSFORM:TRANSMIT
    src         REG:END-NC-LINKAGE1)
    dst         REG:END-NC-PIVOT1)
    dimr        (ALONG-DEPTH NEG)
    from        0
    to          ST:FORCE-NC-PIVOT-END1)
```

In this example, the nutcracker components O:NC-LINKAGE1 and O:NC-PIVOT are in contact in the dimension and direction (ALONG-DEPTH NEG). The linkage handle, REG:HANDLE-NC-LINKAGE1, when pressed, produces an applied force at the same location and in the same (ALONG-DEPTH NEG) dimension and direction. In EDISON, when a force is applied to an object, the force is conveyed to all regions. That is, no transient effects are modeled in EDISON. The *src* role of BPP-TRANSFORM:TRANSMIT is filled by the region REG:END-NC-LINKAGE1, because it is O:NC-LINKAGE1's only regional location which is connected to O:NC-PIVOT. The *dst* role is similarly filled by the region REG:END-NC-PIVOT1. The *from r*ole is filled by a state which represents the O:NC-LINKAGE1 force in *dimr* (ALONG-DEPTH NEG) prior to the process, and the *to r*ole is filled by a state which represents the force transmitted by REG:END-NC-LINKAGE1 to the pivot piece. When the same interactions are described for O:NC-LINKAGE2 and O:NC-PIVOT, a force of the same magnitude is transmitted to REG:END-NC-PIVOT2, but in the (POS) direction. When the global force state on O:NC-PIVOT is updated, these two forces cancel one another.

2.4.1 Behavioral Process Primitives (BPPs)

BPP-TRANSFORM is called a behavioral process primitive (BPP). In EDISON, there are five BPPs for describing mechanical behavior: BPP-MOTION, BPP-RESTRAIN, BPP-TRANSFORM, BPP-STORE, and BPP-DEFORM. Behavioral primitives are distinguished by the type of state they produce, and are causally related as illustrated in Fig. 3 and Table 1.

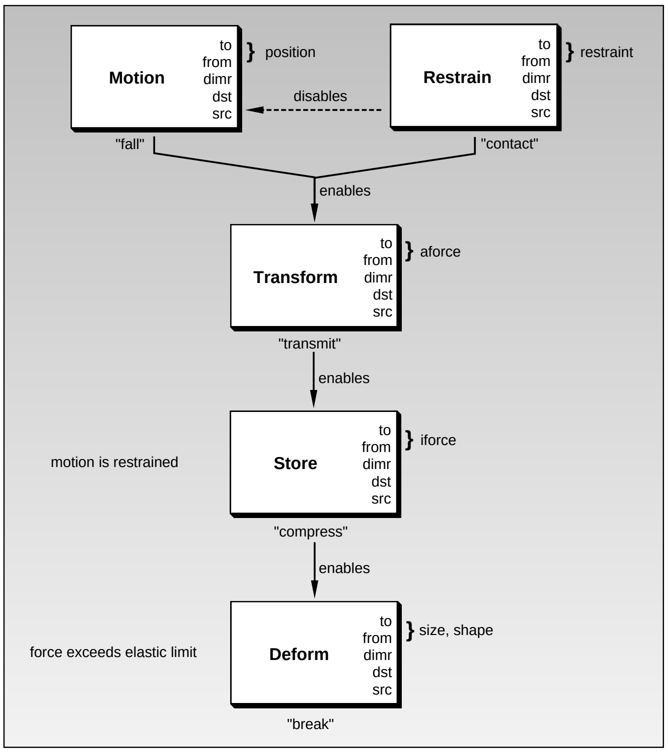


Figure 3: Behavioral process primitives. Each primitive type is illustrated as a process. Centered in the process is the primitive type. On the inside edges are the process roles. Representative enabling preconditions are illustrated to the left of each primitive. The physical state resulting from an enabled primitive is illustrated outside the right side of each process. Beneath each process is an specialization example for the primitive.

BPP-MOTION describes object motion and causes changes in physical proximity. BPP-RESTRAIN describes object connectivity and causes changes in physical restraint. BPP-MOTION produces observable states, whereas BPP-RESTRAIN produces states which aren't always observable, however, BPP-RESTRAIN can disable BPP-MOTION. Consequently, BPP-RESTRAIN dependencies can enable more complex behavioral interactions. BPP-TRANSFORM describes force transformation between objects and causes changes in force dimension, direction, and magnitude. In applied mechanics, the field of rigid-body statics describes force transformations which do not modify the object. BPP-TRANSFORM therefore describes qualitative rigid-body statics. BPP-TRANSFORM is enabled by an applied force and BPP-RESTRAIN (nominally, BPP-RESTRAIN:CONTACT). BPP-STORE describes non-permanent (elastic) deformation, and causes changes in internal energy. BPP-STORE is enabled by an applied force, BPP-TRANSFORM (nominally, BPP-TRANSFORM:TRANSMIT), and a relative restraint state on the object. BPP-DEFORM describes permanent object deformation and causes changes in physical size or shape. When an object is deformed elastically, such that its original size and shape are retained, BPP-STORE is implied. If the elastic limit is exceeded, as when an object is bent or broken, BPP-DEFORM is implied.

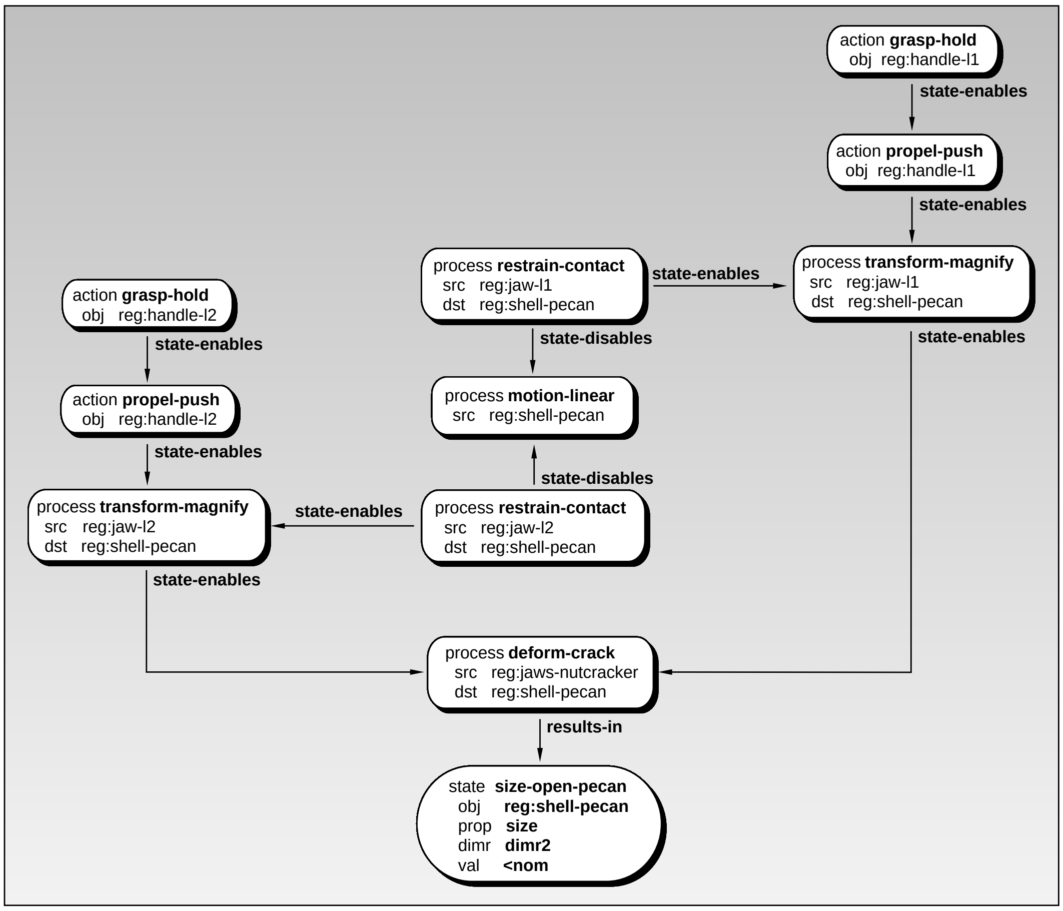
Process	Behavioral Preconditions	Quantity Preconditions	Causal State
BPP-RESTRAIN	none	Equivalent Region Position	restraint
BPP-MOTION	BPP-RESTRAIN (disables)	Nonzero Global Force	proximity
BPP-TRANSFORM	BPP-RESTRAIN, BPP-MOTION	none	force or motion
BPP-STORE	BPP-TRANSFORM	Applied Force < Elastic Limit	elastic size, shape, and energy
BPP-DEFORM	BPP-STORE	Applied Force > Elastic Limit	plastic size and shape

Table 1: Behavioral process primitives, their behavioral and quantity preconditions, and resulting state types.

BPPs describe dynamic behavior; they are enabled and disabled by physical states, and they cause physical state changes. Physical states can be caused by actions, or other processes, and can describe object characteristics. These different state types describe two precondition types for BPP enablement: *behavioral preconditions* and *quantity preconditions*, respectively.

2.4.2 Causally Sequenced BPPs

Complex object behavior is represented by combining BPPs in causal sequences. Consider the behavioral sequence associated with cracking pecans open with a nutcracker, illustrated in Fig. 4.



action **grasp-hold** obj reg:handle-l1
state-enables
action **propel-push** obj reg:handle-l1
state-enables
process **restrain-contact** src reg:jaw-l1 dst reg:shell-pecan
state-enables
process **transform-magnify** src reg:jaw-l1 dst reg:shell-pecan
action **grasp-hold** obj reg:handle-l2
state-enables
state-disables
state-enables
action **propel-push** obj reg:handle-l2
process **motion-linear** src reg:shell-pecan
state-enables
state-disables
process **transform-magnify** src reg:jaw-l2 dst reg:shell-pecan
state-enables
process **restrain-contact** src reg:jaw-l2 dst reg:shell-pecan
state-enables
process **deform-crack** src reg:jaws-nutcracker dst reg:shell-pecan
results-in
state **size-open-pecan** obj **reg:shell-pecan** prop **size** dimr **dimr2** val **<nom**

Figure 4: BPP sequences for representing nut cracking with a nutcracker. The proximity states which enable BPP-RESTRAIN:CONTACT-JAW have been omitted. The storage of elastic energy in nc-spring (BPP-STORE:COMPRESS), and dimensions and values are not shown. The state-enables link name is a notational shorthand for X results-in STATE, STATE enables Y.

Holding and pressing the nutcracker handles are actions which produce an applied force state on the nc-linkages. The sequence shown assumes the pecan is positioned between the nc-linkage jaws, enabling BPP-RESTRAIN:CONTACTs at the nc-linkage jaw regions. The force and restraint states jointly enable force transmission to the pecan shell at both nc-linkage jaw regions. Because the nc-linkages are pivoted at one end (not represented in this figure[1]), a BPP-TRANS-

FORM:MAGNIFY is enabled, and the force transmitted is magnified at the pecan shell. If the magnified force exceeds the pecan shell elastic limit, then a BPP-DEFORM:CRACK will be enabled. The state caused by BPP-DEFORM is a change in size. This state is observable and is called the pecan open state.

Representing device behavior in BPP sequences is a detailed method of expressing nutcracker function in opening pecans. In representing the BPP sequence, many physical states are made explicit which might not be observable to a human nutcracker user. Using a black box approach, the same behavior can be described using only the observable states. This functional approach is significant because devices are used in actor plans to achieve observable (desired) states.

Naive Mechanics describes device use and function, and device behavior is used during experimentation, or to diagnose and explain devices which fail to function as expected. The degree of reasoning captured by EDISON processes is intentionally limited to their behavioral and quantity preconditions, along with the physical states they produce. In this respect, EDISON BPPs are both similar to and simplified versions of the qualitative processes introduced by Forbus, [Forbus, 1983]. Nevertheless, BPPs and BPP sequences form the backbone for representing device function, and these reasoning capabilities are necessary to support the inferences and predictions needed to reason about situational device use.

2.5 Device Function

A device may be associated with many functions, depending on the contexts in which it is applied. For example, a crank can opener may be used to open cans, but it may also be used to hold down papers, to crack open nuts, or pound nails. The same can opener is used in each application, but the environment, and how the device is applied, change.

In EDISON, a device *function* is a black box representation of a BPP sequence. A function is represented as a frame: it has roles for the object regions necessary to describe the outward behavior, it has entry conditions which enable the BPP sequence, and it has exit states resulting from the BPP sequence. A device function has three roles: an *appl*, a *pivot*, and a *react*. The function *appl* role is instantiated by the region where the force, which initiates the function, is applied. The *pivot* role is instantiated by the region which acts as a pivot, if one exists. The *react* role is instantiated by the region where the function is reacted. A function has three types of entry conditions: *applied preconditions*, *behavioral preconditions*, and *physical preconditions*. Applied preconditions (AP)

[1] Process representation has not been presented in detail here; see [Hodges, 1992] for a complete discussion of all EDISON knowledge structures and their interactions.

are the forces required to initiate an object function. Behavioral preconditions (BP) are states resulting from behavioral interactions (e.g., connectivity) with other objects. Physical preconditions (PP) describe the object properties and regions required to instantiate the function roles. The applied, behavioral, and physical preconditions all separately enable device function. Each is a necessary, but not sufficient condition.

2.5.1 Machine Primitives (MPs)

Some objects are associated with a single function. These object/function pairs are called *machine primitives* (MPs), and are introduced to represent and reason about complex mechanical function. There are eleven MPs, each associated with a simple object: MP-LINKAGE, MP-LEVER, MP-WHEEL-AXLE, MP-GEAR, MP-PULLEY, MP-BEARING, MP-PLANE, MP-BLADE, MP-SCREW, MP-SPRING, and MP-CONTAINER. Figure 5 illustrates the EDISON MPs in a hierarchy organized by the physical differences between MP objects, and hence according to function similarity and specialization.

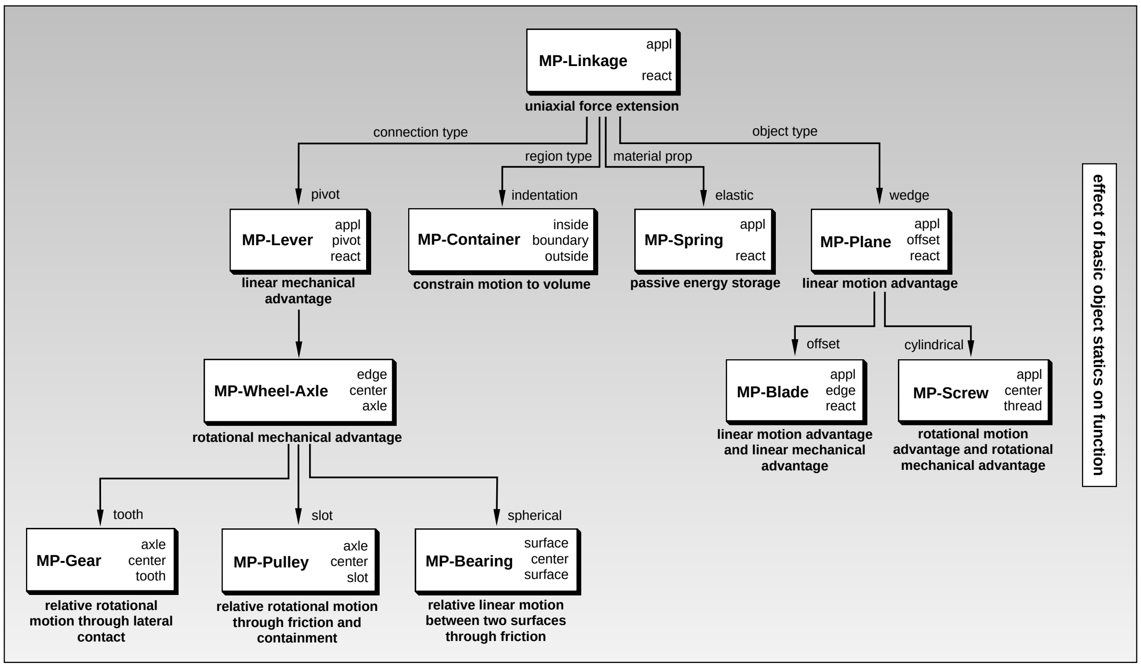

MP-Linkage appl react
uniaxial force extension
connection type
region type
material prop
object type
pivot
indentation
elastic
wedge
MP-Lever appl pivot react
linear mechanical advantage
MP-Container inside boundary outside
constrain motion to volume
MP-Spring appl react
passive energy storage
MP-Plane appl offset react
linear motion advantage
offset
cylindrical
MP-Blade appl edge react
linear motion advantage and linear mechanical advantage
MP-Screw appl center thread
rotational motion advantage and rotational mechanical advantage
MP-Wheel-Axle edge center axle
rotational mechanical advantage
tooth
slot
spherical
MP-Gear axle center tooth
relative rotational motion through lateral contact
MP-Pulley axle center slot
relative rotational motion through friction and containment
MP-Bearing surface center surface
relative linear motion between two surfaces through friction
effect of basic object statics on function

Figure 5: Machine primitives (MPs). Each MP is depicted with its appl, pivot, and react roles associated with the physical object regions which instantiate them. An example of the object type associated with each MP is shown below it.

Each MP instantiates a device function, so each MP object has regions which instantiate the function appl, pivot, and react roles. The simplest device function is one which transmits force (BPP-TRANSFORM:TRANSMIT) from one object to another. MP-LINKAGE associates this function with linkage objects. For example, a stick can be used to extend and probe by transmitting force, and MP-LINKAGE can represent the stick and its function. The MP-LINKAGE *appl* role is instantiated with one stick end region, and the MP-LINKAGE *react* role is instantiated with the other stick end region. By adding a fulcrum or pivot region to a linkage object, leverage can be described with MP-LEVER, which represents the function of magnifying or reducing applied force (see Table 2). When rotational force is transmitted radially, or translated along the pivot axis, the function/object is represented with MP-WHEEL-AXLE. If the wheel object's react region is toothed, then the function/object is represented with MP-GEAR. If the wheel object's react region is slotted, then the function/object is represented with MP-PULLEY. The primitive MP-BEARING is used to transmit motion of other objects, along its outer surface, so the bearing object's appl and react regions describe the same surface. The resulting bearing object must be cylindrical or spherical.

A linkage object with an indentation is capable of containment, and represented with the primitive MP-CONTAINER. A container-object has an inside surface associated with the indentation, an outside surface, and an edge or boundary. The edge and outside regions correspond to the MP-LINKAGE appl and react regions (e.g., one could stand on a bowl). MP-SPRING is used to represent the function/object of passive energy storage. Any object which is perturbed with force value large enough to enable the process BPP-STORE is represented with MP-SPRING.

Wedge-shaped linkage objects are used to magnify or reduce the force applied in one dimension with the force reacted in another dimension. The wedge shape provides a built-in pivot in the form of the wedge offset angle. These function/object types are represented with MP-PLANE. When the plane object is applied primarily along its edge region, instead of at its react region, to enable BPP-DEFORM processes, then its function is represented with the primitive MP-BLADE. The plane object can be used linearly or rotationally. When used rotationally, the function/object is represented with MP-SCREW.

Primitive	**Process**	**MP Object Characteristics**	**I/O Regions**
MP-LINKAGE	TRANSFORM:TRANSMIT	Material: Elastic in dimr	Appl, React
MP-LEVER	TRANSFORM:MAGNIFY	Add Pivot Region	Appl, Pivot, React
MP-WHEEL-AXLE	TRANSFORM:MAGNIFY	Shape: Circular	Edge, Center, Axle
MP-BEARING	TRANSFORM:TRANSMIT	Shape: Spherical or Cylindrical	Surface, Center, Surface
MP-GEAR	TRANSFORM:MAGNIFY	Add Tooth Region; Parts: Gear or Chain	Tooth, Center, Axle
MP-PULLEY	TRANSFORM:MAGNIFY	Add Slot Region; Parts: Pulley and Belt	Slot, Center, Axle
MP-PLANE	TRANSFORM:MAGNIFY	Shape: Wedge (Offset Angle)	Appl, React
MP-BLADE	DEFORM:CUT	Attribute: Sharp	Appl, Edge, React
MP-SCREW	TRANSFORM:MAGNIFY	Shape: Cylindrical	Appl, Center, Thread
MP-SPRING	STORE	Material: Elastic in dimr	Appl, React
MP-CONTAINER	RESTRAIN:CONTACT	Add Indentation Region	Inside, Boundary, Outside

Table 2: Machine primitive (MP) classification.

In Table 2, a process describes the BPP class which a machine primitive enables. MP object characteristics describe the physical requirements for the MP object. MP-LINKAGE can be instantiated by any linkage object. MP-WHEEL-AXLE need not be circular, as described in the table. The wheel-axle object is generally associated with circular objects, although a crank, which is not circular in general, is represented with MP-WHEEL-AXLE. The gear object associated with MP-GEAR is a combination of objects in tooth-contact. Likewise, the primitive MP-PULLEY is enabled for a pulley object consisting of a pulley and belt in surface contact. Both MP-GEAR and MP-PULLEY are used to transmit force from their appl region to their react region, if the regions are the same. The same primitives are used to transform force from their appl region to their react region, if the regions are not the same. A pinion gear is an example of the former, while a gear and shaft arrangement is an example of the latter.

2.5.2 MP Specialization

Machine primitives can be specialized by the physical characteristics of their objects. For example, there are three MP-LEVER types (and MP-WHEEL-AXLE types, as implied by Fig. 5), defined by the relative location of the appl, react, and pivot regions: (a) teeter-totter-like, (b) wheel-barrow-like, and (c) crane-like. These lever types are commonly referred to as Type 1, Type 2, and Type 3, respectively, in the literature (e.g., [Macaulay, 1988]), as described below and illustrated in Fig. 6.

MP-LEVER:TYPE1 (*Teeter-Totter*): Applied and reacted forces are on opposite sides of the pivot. Forcing regions move in opposite directions. Force is magnified.

MP-LEVER:TYPE2 (*Wheel-Barrow*): Reacted force is located between the applied force and the pivot. Both forcing regions move in the same direction. Force is magnified.

MP-LEVER:TYPE3 (*Crane*): Applied force is located between the reacted force and the pivot. Both forcing regions move in the same direction. Motion is magnified.

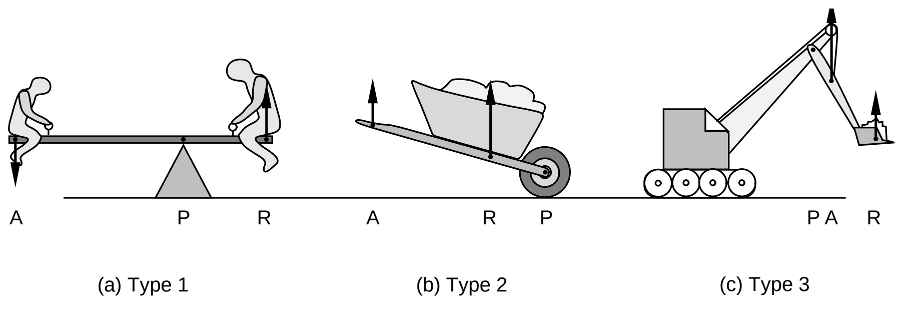


Figure 6: Three lever device classes: (a) a Type 1 lever, (b) a Type 2 lever, and (c) a Type 3 lever. The A, P, and R beneath each device represent the relative positions of appl, pivot, and react regions in the lever types, respectively. In (b) and (c), the use of the machine dramatically changes, since (c) has a force magnification less than unity. However, as with a human arm, much larger distances can be traversed with this design.

In Fig. 6, the arrows are shown at the appl and react region locations for their respective devices. The arrow length provides a measure of relative force magnitude values. The value of reacted force is related to the applied force by comparing the relative appl-to-pivot and react-to-pivot lengths. In EDISON, the result takes one of three values: less than nominal, nominal, or greater than nominal (<NOM, NOM, >NOM), where nominal is the applied force value. For example, if AP is the length from the appl region to the pivot region, and PR is the length from the react region to the pivot region, then the ratio AP:RP in Fig. 6a is >NOM. The reacted force value is greater than the applied force value. The same is true of Type 2 levers, and the opposite is true of Type 3 levers.

2.5.3 Nutcracker Functions and Machine Primitive Instantiation

Machine primitives can be used to represent different device functions. The nc-linkages of a nutcracker, without a nut, can be used to hammer and probe. In this capacity they instantiate MP-LINKAGE. Leverage is generated when the nutcracker nc-linkages are pressed together when a pecan is placed between them. The contact region between the nc-linkages and pecan instantiates MP-LEVER:TYPE2 and force applied at the handles is magnified at the nut. The hierarchical relationship between linkage and lever objects enables MP-LINKAGE and MP-LEVER to be instantiated in the same device representation.

2.5.4 MP Combination and Mechanical Devices

Two machine primitives can be combined as long as their preconditions are satisfied. Complex device functions can be represented by combining MPs like Tinker Toy pieces. Consider the crank can opener (cco) illustrated in the upper part of Fig. 7. The cco can be used to cut metal cans, pound nails, crack nuts, and hold down papers. The cco consists of eight components: two linkages, a rivet, a crank with axle, two gears (one with axle), and two disks. Each cco component is represented by an MP, as illustrated in the lower part of Fig. 7. The linkages are represented with MP-LEVERs because of the common rivet, and each enables force magnification. The crank and axle are represented with MP-WHEEL-AXLE, which enables rotational force magnification and transmission to the gear and disk also using the common axle. The gears and toothed disk are represented with MP-GEAR, which enables position control and force transmission. Finally, the cutting disk is represented with MP-BLADE, which enables cutting. Thus, every machine related cco use is represented by the MPs which comprise the device.

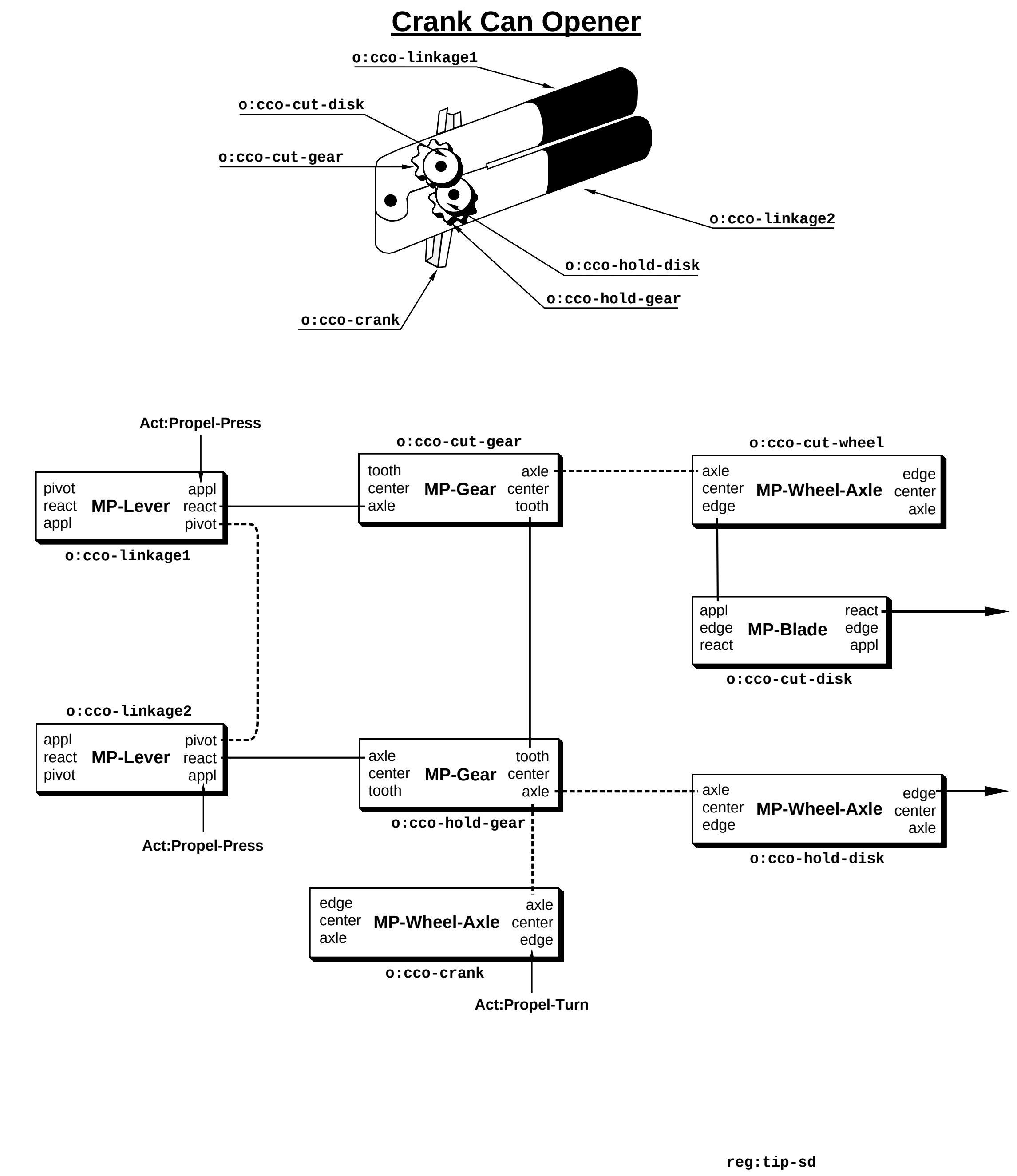

Crank Can Opener
o:cco-linkage1
o:cco-cut-disk
o:cco-cut-gear
o:cco-linkage2
o:cco-hold-disk
o:cco-hold-gear
o:cco-crank
Act:Propel-Press
o:cco-cut-gear
o:cco-cut-wheel
pivot react appl MP-Lever appl react pivot
o:cco-linkage1
tooth center axle MP-Gear axle center tooth
axle center edge MP-Wheel-Axle edge center axle
appl edge react MP-Blade react edge appl
o:cco-cut-disk
o:cco-linkage2
appl react pivot MP-Lever pivot react appl
axle center tooth MP-Gear tooth center axle
o:cco-hold-gear
axle center edge MP-Wheel-Axle edge center axle
o:cco-hold-disk
Act:Propel-Press
edge center axle MP-Wheel-Axle axle center edge
o:cco-crank
Act:Propel-Turn
reg:tip-sd

Figure 7: Crank can opener (cco) components (upper) and MP composition (lower). Dotted lines represent shared roles across object boundaries. The actions illustrated represent the nominal application of cco can-opening function, cut-open. O:CCO-LINKAGE1 and O:CCO-LINKAGE2 have been represented as MP-LEVERs to simplify the diagram. Thick lines with arrows represent regions where the function is reacted.

Figure 7 shows functional dependency and flow of force between cco MPs. This representation doesn't specify a particular interpretation of cco function. For example, the three actions associated with the (nominal) CUT-OPEN function: ACT:PROPEL-PRESS at O:CCO-LINKAGE1 and O:CCO-LINKAGE2, and ACT:PROPEL-TURN at O:CCO-CRANK, are shown. No mention is made of a can or how the actions are applied. This information is part of an OPEN(CAN) plan. Thus, if another object is placed between the handles, and they are pressed together, the individual lever object functions can be effected on the object without consideration of the other cco MPs.

2.6 Device Use in Problem-Solving Situations

Objects can be used to achieve different goals in different situations. In EDISON, object use is defined as the application of an object function in the context of some goal and environment. Each use is associated with a particular function application. By defining the relationship between use and function, a determination of which device and function are applicable in a given context can be made. A plan in which a device function is applied is called a *device-use* plan. The device and its functions are causally related by MP applied, behavioral, and physical preconditions. The plan and device are causally related by the context in which the device function is applied. The functions which define object behavior are not dependent on the surrounding context in which an object is applied, but the plans through which functionality is effected *are*. To improvise solutions to real problems, a problem solver must be able to recognize object utility in the planning context. Choosing between different objects, and their functions, according to context, requires a causal interaction between plans and functions.

2.6.1 Device-Use Plans

The device-use plan used in EDISON is similar to Schank's [Schank & Abelson, 1977] USE(object) plan. Each component on the right-hand side of USE(object), except DO, represents a subgoal to be achieved on the way to the plan's enablement. DO represents the action taken by an intentional agent once the subgoals have been achieved. In the Schank and Abelson model, d-goals and i-goals are called delta goals, and instrumental goals, respectively, and support the achievement of higher- level goals.

USE(object) = FIND(object) + D-CONT(object) + I-PREP(object) + DO

Device-use is a specialization of USE(object), where object control (D-CONT) and preparation (I-PREP) involve device function, and where DO is a form of the PROPEL action which initiates the function by causing an applied force on it. For example, the use of a nutcracker to open pecans requires the user to grasp the nutcracker handles, spread them apart, place the pecan in the nutcracker jaws, close the handles, and press them together. Grasping satisfies the subgoal D-CONT. Spreading, placing and closing satisfy the subgoal I-PREP. Pressing initiates the function MP-LEVER.

2.6.2 Motivation for Device Application

The **Nutcracker** situation does not begin with the use of the nutcracker, but, rather, with a goal to open pecans. Goals which require changes in physical state are related to mechanical behavior. The

D-CONT(pecan) goal motivates a subgoal to open the pecan, which requires a change in the pecan restraint states imposed by its containment. In EDISON, each BPP is associated with a *behavioral delta goal*: D-MOTN, D-REST, D-FORC, D-STOR, and D-SIZE. The pecan restraint state change needed in **Nutcracker** is represented with delta-restraint (D-REST).

Device functions are often applied when a problem solver is incapable of producing the changes in physical state associated with behavioral delta goals. For example, in order to crack a pecan shell open, a device capable of producing the force necessary to crack the shell can be used. Finding a device capable of mechanical advantage motivates a class of goals, called *mechanical instrumental goals*., which relate a desired function to the devices which can instantiate it. There are four mechanical instrumental goal types in EDISON: I-REACH, I-MECH, I-CNTN, and I-CLIMB. In **Nutcracker**, an instrumental goal to produce mechanical advantage (I-MECH) is motivated by a plan to crack the pecan open.

Behavioral delta goals represent the causal interactions between high-level goals, planning failures, and device use. Mechanical instrumental goals represent the causal interactions between device-use plans and device functions. These goals provide a means to explicitly represent the MPs a device might instantiate with the types of goals and plans in which the device might be used. They do not provide a means for choosing a device in a particular problem-solving context.

2.7 Functional Similarity and Improvisation

Nutcracker is a situation in which the device normally used is unavailable. In such cases, the problem solver can try another plan or find a suitable replacement. The latter solution may require *improvisation*, which is the application of a device not designed for the intended use. Finding alternatives requires some method for evaluating device suitability. In real situations, object use depends on the satisfaction of two types of constraints: (1) the *functional constraints* which enable MP function, and (2) the *contextual constraints* which support plan application.

2.7.1 Functional Constraints and MP Equivalence

An object function can be applied to a situation if the applied, behavioral, and physical preconditions which enable MP function are satisfied. For example, consider the cco as a potential device in the **Nutcracker** situation. Why would a problem solver (or EDISON) choose this device for cracking nuts? Whatever object is chosen must satisfy two MP requirements which the nutcracker satisfies: it must be capable of transmitting forces from itself to the nut, and it must be strong enough to transmit the forces required to break the nut without itself breaking. These requirements fall into two categories: *regional constraints* and *property constraints*.

1. Regional Constraints: An object can instantiate an MP if it has regions which instantiate those required by the MP. A nutcracker is a generalized MP-LEVER, and has appl, pivot, and react regions. The cco also has lever object regions in its O:CCO-LINKAGE1 and O:CCO-LINKAGE2 components. Objects which instantiate the regions of the same MP are r*egion equivalent*. The nutcracker and crank can opener are region equivalent for the machine primitive MP-LEVER:TYPE2.
2. Property Constraints: An object can instantiate an MP if its property values enable the behavior associated with the MP. The nutcracker breaking strength is much greater than that of a pecan shell. The cco can be used as a nutcracker because its breaking strength is comparable to that of a nutcracker. Objects which have the same property values are *property equivalent.* The nutcracker and cco are property equivalent for the process BPP-DEFORM:CUT-CRACK when applied to MP-LEVER:TYPE2. BPP-DEFORM:CUT-CRACK is enabled because the breaking strength of the object being deformed is less than the force produced by the lever object.

When two devices are region and property equivalent with respect to a particular device function, then they instantiate an MP's preconditions, and they are *MP equivalent*. MP equivalence is a similarity metric for device function based purely on physical characteristics, without consideration of a use context.

2.7.2 Contextual Constraints: Property Attributes

Each situation imposes constraints on the interpretation and choice of devices during problem solving. A device can be applied to a situation if those properties required by a particular device function comply with the physical constraints of the situational context. Devices meeting these criteria are *contextually appropriate*.

Consider a claw hammer used for prying in two contexts. A claw hammer used to pry nails from wood is dependent on two properties: size and breaking strength. The claw size must enable it to be placed between the nail head and the wood. The breaking strength determines whether the claw and nail will move, or whether the nail head, claw or wood will be damaged. For prying nails from wood, the claw hammer is contextually appropriate, because its physical property values enable the prying function and match the size and breaking strength constraints of the nail and wood.

The claw hammer is not contextually appropriate for prying a lid off a paint can. In this case, the same two properties affect the choice of device used for prying. The size property values determine whether the claw will fit the container slot, while the breaking strength values determine whether or not the can lid will be removed before something breaks. The claw hammer cannot be used for opening a paint can, because the claw is too wide to fit the slot between the can lid and can vessel. Thus the claw hammer, which is designed for use as a prying device, and which is MP equivalent to a screwdriver used to pry a lid from a varnish can, fails to meet the physical constraints (i.e., can slot width) of the situation.

Context constrains device choice in problem solving by enabling or disabling potential device functions. MP behavioral preconditions are dependent on the values of other objects' physical properties. In the example just presented, the sizes of the hammer claw and can slot must be known before a determination can be made whether the hammer will work as a prying device. A human problem solver, however, is unlikely to know the exact size of the hammer claw or can slot. On the other hand, to pry a container open, a human problem solver knows that a device capable of prying is needed, and that the device chosen must *fit* the can slot. Making these determinations requires knowledge of how the claw hammer's size property values *compare* to those of the can slot.

Both contextual and object property constraints on object use are affected by property comparisons. For example, when choosing an object for prying, we tend to look for objects that are strong rather than objects with a specific strength. The object's strength causally affects object function, but the term "strong" causally affects plan application, because the term is derived from a problem-solving context. Unfortunately, contextual comparisons of width and strength do not have one-to-one correspondences with the associated property values (size in width, and breaking strength, respectively). An object which is strong in one context may not be in another. "Wide" and "strong" are useful for planning, because they include a wider range of potentially applicable objects; however, their use cannot be guaranteed outside the context in which any particular object attains its "wide" or "strong" designation.

In EDISON, property value comparisons are represented with property *attributes*. An attribute represents a qualitative comparison between the property values of two devices in a particular context. Attributes have five roles: *obj*, *situ*, *prop*, *ref*, and *val* (<, =, >). The attribute representation for a "strong" nutcracker, used for cracking open a pecan, is shown below.

```
(attribute ATT:STRONG-NC-LINKAGE1
  obj       O:NC-LINKAGE1
  situ      S:NUTCRACKER
  prop      BREAKING-STRENGTH
  ref       (state ST:BSTRENGTH-PECAN-SHELL
              obj     REG:SHELL-PECAN
              prop    BREAKING-STRENGTH)
  val       >)
```

ATT:STRONG-NC-LINKAGE1 is a plan enablement for nutcracker use in **Nutcracker**. Its meaning is that the nutcracker has a breaking strength greater than that of the nutshell. The *obj* role describes the object with which the attribute is associated, namely the body of O:NC-LINKAGE1. The *situ* role associates the object with a particular goal/plan context. Context must be included in the representation for property value comparisons, because a device may not have the same relative value when the context changes. The *prop* role describes the property of the device being represented. The *ref* role describes the reference region in a quantity space where the comparison is valid. In this case, the reference value is the pecan shell's breaking strength. Changing the filler for the *ref* role changes the meaning of the attribute.

2.7.3 Problem-Solving Situations

If two devices are MP equivalent and contextually appropriate, then they can perform the same task under the same conditions. Devices which satisfy these criteria are *functionally equivalent*. Problem solving is experiential. Human problem solvers use experience to evaluate and solve problems. The actions they perform and the objects they use can later be remembered along with the context. In EDISON, problem-solving experiences and scenarios are represented with a knowledge structure called a *situation*. A situation describes device application in the context of a goal and a set of environmental constraints. Environmental constraints are represented as a collection of physical states, some of which may describe subgoal violations.

The **Nutcracker** situation illustrates the effect of environmental state on device-use plans. An object is closed and must be opened. The closed state is an environmental constraint which violates a D-CONT subgoal supporting a higher-level I-PREP (pecan) goal. This violated subgoal motivates a D-REST goal to open the object. The D-REST subgoal can be achieved with a number of plans, but the plan chosen determines which, if any, device and device function is applied.

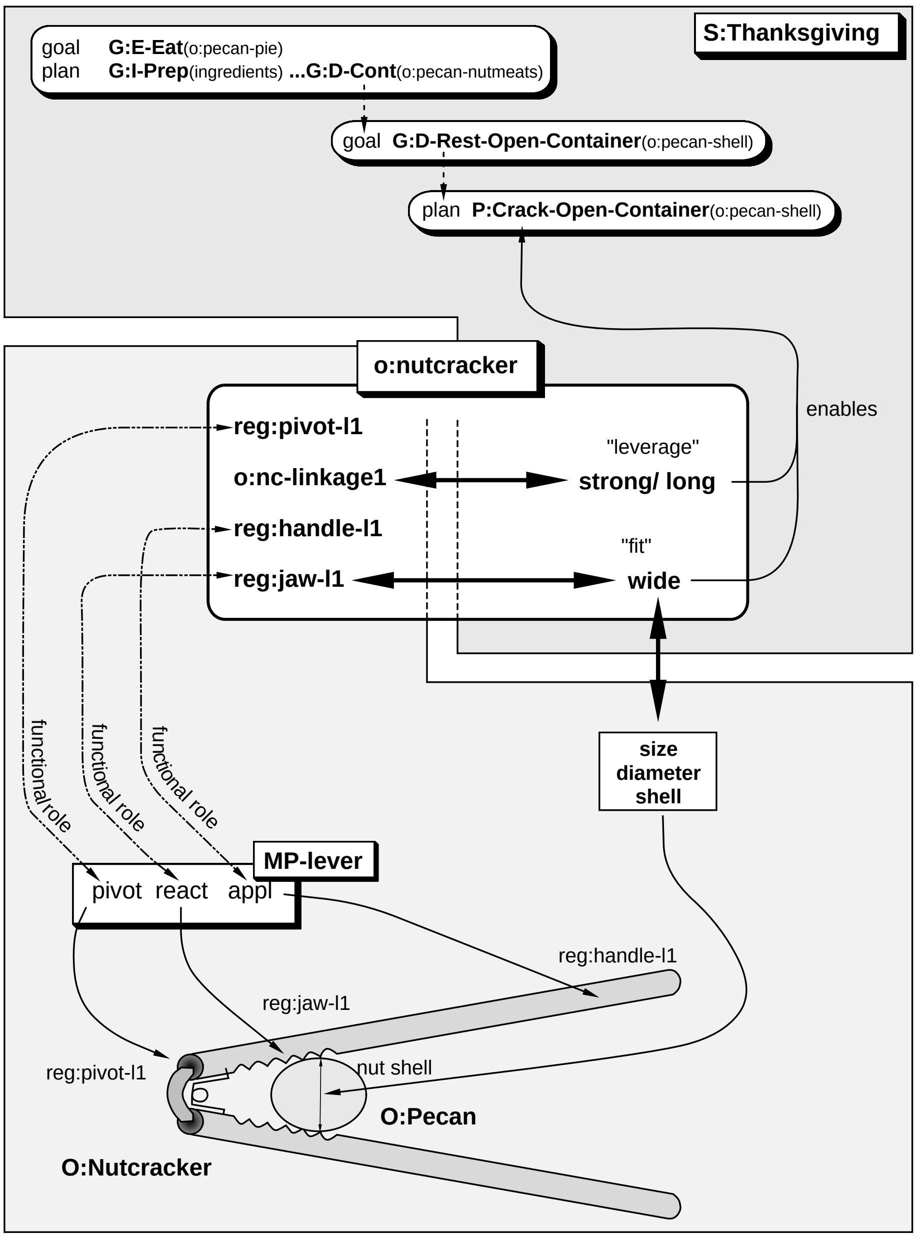

S:Thanksgiving
goal G:E-Eat(o:pecan-pie)
plan G:I-Prep(ingredients) ...G:D-Cont(o:pecan-nutmeats)
goal G:D-Rest-Open-Container(o:pecan-shell)
plan P:Crack-Open-Container(o:pecan-shell)
o:nutcracker
enables
reg:pivot-l1
o:nc-linkage1
"leverage"
strong/ long
reg:handle-l1
"fit"
reg:jaw-l1
wide
size
diameter
shell
functional role
functional role
functional role
MP-lever
pivot react appl
reg:handle-l1
reg:jaw-l1
reg:pivot-l1
nut shell
O:Pecan
O:Nutcracker

Figure 8: The **Thanksgiving** situation. The upper block depicts the goal/plan level. The middle block depicts the effect of context on the choice of object, and the nutcracker regions which context affects. The lower block is a pictorial representation of how the nutcracker instantiates the regional enablements (i.e., appl, pivot, and react) for MP-LEVER.

Figure 8 illustrates a situation, **Thanksgiving**, where a nutcracker is used to open a pecan. The goal, plan, device, and function representations are illustrated in the figure. The high-level goal to prepare a pecan pie for Thanksgiving has a subgoal of obtaining the pecan nut meats. Because the nut meats are in their closed shells, the D-CONT subgoal is violated. A new subgoal to open the shells, D-REST-OPEN-CONTAINER, is motivated, and a plan for cracking the shells open, CRACK-OPEN-CONTAINER, is selected. The plan is enabled by devices which instantiate MP-LEVER, via the I-MECH goal. The nutcracker regions REG:HANDLE-L1, REG:JAW-L1, and REG:PIVOT-L1 instantiate the roles of MP-LEVER, thus satisfying the I-MECH subgoal and enabling the CRACK-OPEN-CONTAINER plan. The nutcracker satisfies the contextual constraints, because the pecan fits between O:NC-HANDLE1 and O:NC-HANDLE2. The attribute ATT:WIDE-NC, which describes the nutcracker jaw width with respect to the pecan diameter, enables CRACK-OPEN-CONTAINER.

3.0 Summary and Conclusions

In this article three representation levels are presented, each supporting a different aspect of mechanical problem solving. The lowest level is concerned with individual objects and their composition in devices. An individual object is represented by its property values and its regions. The second level is associated with state change. Changes in an object's property values describe its behavior, and its regions determine where behavioral interactions take place. Five behavioral primitives account for the types of state change which mechanical objects can exhibit. These primitives can be combined to represent complex behavior. Behavioral sequences which produce observable behavior are associated with object function. Eleven machine primitives associate a physical object with an expected behavioral sequence. MPs are combined to represent complex mechanical functions. The third level is associated with device use. A device may have many functions, each associated with a particular goal/plan context. The applicability of a device in a particular situation depends on whether the device is functionally equivalent to devices which have been used in the past. In order to distinguish between device functions, a set of device-related goals were introduced which associate device function with the plans in which it is applied. Device attributes relate the objects in a particular context to the plans in which they can participate.

The EDISON approach addresses the general problem-solving task in two ways. First, the introduction of BPPs and MPs provides a finite set of constructs for representing mechanical devices. The BPP level of representation is useful for diagnosing machine malfunction. The representation of a device with MPs enables it to be applied to tasks for which it was not originally intended, and allows for comparisons to other devices, based on shared MPs. Second, the interaction of function and use supports the choice of devices and device functions based on how they will be used, and in which context.

4.0 Related Computational Approaches

Inspiration for the EDISON project was drawn from the following areas of computational modeling:

Qualitative Reasoning: My approach to representing object behavior using qualitative processes is built on the QP theory introduced by Forbus [Forbus, 1983]. The intended domains of QP theory and BPPs overlap. BPPs are not intended to support detailed reasoning, but to provide some simulation and explanation for a system designed to reason about object use and function. BPPs cannot perform detailed diagnosis; there is no internal representation for the so-called physical laws. For example, when an object stops moving, it either ceased to be forced or its path was somehow blocked. This level of reasoning is basically naive, which is the intended scope of the project.

Functional Reasoning: My approach to representing object function is similar to the functional representation introduced by Sembugamoorthy and Chandrasekaran [Sembugamoorthy, 1986]. There are two differences between the models. First, I am currently focusing my efforts on a model which can be used for purely mechanical systems. Second, my model describes all mechanisms with a discrete, rather than extensive, set of function primitives. My claim is that MPs completely describe mechanical function so that no expressiveness is lost.

Intentional Reasoning about Objects: My approach in representing the ways people use objects in solving problems parallels the approaches of Rieger, [Rieger, 1985], and Lehnert, [Lehnert, 1978]. My approach to object use is based on actor goals and plans; however, the qualitative model is motivated by consistency with engineering physics.

Case-Based Reasoning in Physical Systems: Other researchers share my enthusiasm for approaching engineering problems using episodic, or case-based, reasoning ([Goel, 1989], [Hammond, 1989], and [Navinchandra, 1989]). The primary differences have been in domain. However, the EDISON project was originally intended to address mechanical creativity, and an episodic approach was mandated from the outset. As a result, my representational approach focused on consistency across domain boundaries, and size considerations for large-scale memory models. Regions, BPPs, MPs, and property attributes support the extension to large memory models.

5.0 Current Status and Further Work

The EDISON representation approach has been continually updated with computer models developed to evaluate different aspects of the representation. At present I can represent, recognize, and simulate BPP-MOVE, BPP-RESTRAIN, and BPP-TRANSFORM. These have been used in recognizing mechanical relationships ([Hodges, 1988], [Hodges, 1989]), parsing device descriptions [Dyer, 1987], and mutating devices during experimentation [Dyer, 1986]. The MP reasoning model and problem-solving model are only partially implemented. I currently have a library of twenty-five devices and three problem-solving situations. The processing methodologies used in recognizing and reasoning about objects in my programs are vastly different. When I began the EDISON project, symbolic and rule-based methods were being widely used for both representation and reasoning tasks in AI. I am currently adapting my processing model to combine the access and retrieval advantages of a hybrid semantic network with the processing advantages of a rule-based system, both in the framework of a localist/connectionist computational paradigm [Lange et al., 1989].

References

DeKleer, J. and Seeley Brown, J. (1983): A Qualitative Physics Based on Confluences. *Artificial Intelligence 24*(July 1983), pp. 7-84.

Dyer, M.G., Flowers, M., and Hodges, J. (1986): EDISON: An Engineering Design Invention System Operating Naively. *International Journal of Artificial Intelligence in Engineering 1,* 1 (July 1986), pp. 36-44.

Dyer, M.G., Hodges, J., and Flowers, M. (1987): Computer Comprehension of Mechanical Device Descriptions. Tech. Rep. University of California at Los Angeles, UCLA-AI-87-9, 1987.

Forbus, K. (1983): Qualitative Process Theory. *Artificial Intelligence 24*(July 1983), pp. 85-168.

Goel, A.K. (1989): *Integration of Case-Based and Model-Based Reasoning for Adaptive Design Problem Solving,* Ph.D. dissertation, Ohio State University, 1989.

Hammond, K.J. (1989): *Case-Based Planning: Viewing Planning as a Memory Task,* Academic Press, New York, NY , Perspectives in Artificial Intelligence (1989).

Hodges, J. (1988): *Interactions Between Schema and Text: Recall and Problem-Solving*. 1988, Unpublished research.

Hodges, J. (1989): Device Representation for Modeling Improvisation in Mechanical Use Situations. In *Proceedings of the 11th Annual Meeting of the Cognitive Science Society,* Ann Arbor, Michigan, August 1989, pp. 643-650.

Hodges, J. (1992): Naive Mechanics: A Computational Model for Representing and Reasoning about Simple Mechanical Devices. Ph.D. Dissertation, University of California at Los Angeles, in preparation.

Lehnert, W.G. (1978): *The Process of Question Answering,* Lawrence Erlbaum Associates (1978).

Macaulay, D. (1988): *The Way Things Work,* Houghton Mifflin Company (1988).

Navinchandra, D. and Sycara, Katia, P. (1989): A Process Model of Experience-Based Design. In *Proceedings of the 11th Annual Meeting of the Cognitive Science Society,* Ann Arbor, Michigan, August 1989, pp. 283-290.

Rieger, C. (1985): An Organization of Knowledge for Problem Solving and Language Comprehension. In *Readings in Knowledge Representation*. Morgan Kaufmann, pp. 487-508, 1985.

Schank, R. and Abelson, R. (1977): *Scripts, Plans, Goals, and Understanding,* Lawrence Erlbaum, Hillsdale, NJ , The Artificial Intelligence Series(1977).

Sembugamoorthy , V. and Chandrasekaran, B. (1986): Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems. In *Experience, Memory, and Reasoning*. Lawrence Erlbaum Associates, Kolodner, J. and Riesbeck, C., Ch. 4, pp. 47 - 73, 1986.