#N00014-K-86-0615

2

EDISON Project: Final Report

Michael G. Dyer
Margot Flowers
Jack Hodges

3532 Boelter Hall
Computer Science Department
UCLA, Los Angeles, CA 90024

1 INTRODUCTION

The EDISON project is one of developing computational models capable of creative reasoning and problem-solving with respect to mechanical devices. The overall approach has been that of representing devices as symbolic structures (i.e. a *naive mechanics* approach). The focus of our research has been on representational and processing constructs to support the following cognitive tasks: (1) comprehension of natural language text describing mechanical devices, (2) simulation of device behavior through manipulating symbolic representations of device motions, (3) improvisation through adapting one device for use in an alternate domain, and (4) device invention through mutation of device features. These four tasks serve to test the viability of EDISON device representations.

In carrying out this research, we have been influenced both by the computational and psychological literature.

1.1. Related Computational Approaches

There are four areas of computational modeling from which the EDISON project has drawn inspiration:

(1) *Qualitative physics:* By using symbolic (versus numerical) models, qualitative reasoning systems, e.g. [Hayes, 1978], [DeKleer and Brown, 1983], enable simulations at higher levels of abstraction. Computational reasoning systems utilizing these models have been developed to describe and simulate objects in various physical domains (e.g. fluid-mechanics [Kuipers, 1986], electrical engineering [Granacki and Parker, 1986], and thermodynamics [Forbus, 1983]).

(2) *Natural Language Processing (NLP) for comprehension and invention*: A system cannot be inventive if it cannot understand concepts involving devices, such as their functions, component configurations, and behaviors. There has been some NLP research in understanding the use of objects in narrative settings [Dyer, 1983], [Lehnert, 1978]), and in reasoning about object composition and function [Rieger, 1985], [Wasserman and Lebowitz, 1983]. Natural language processing methods have also been developed to address issues specifically associated with physical descriptions [Lebowitz, 1985] [Granacki and Parker, 1986]. The issue of invention has also been addressed in NLP, through examing the task of story invention [Mueller, 1987] [Turner, 1985].

(3) *Memory Organization for Case-Based Recall:* Device representations reside in an episodic memory and a major aspect of invention is the recall of relevant device components, based on situational and functional indices. Case-based reasoning systems, e.g. [Kolodner, 1984], [Lebowitz, 1980], [Pazzani, 1988], [Hammond, 1989], solve problems by recalling relevant cases and then adapting them to the situation at hand. Thus.the organization and indexing structures of episodic memory are critical for problem solving.

(4) *Automated Discovery*: Systems have been developed to create concepts from first principles, e.g.[Lenat, 1976], [Lenat, 1983], and [Coyne et al., 1987], to perform iterative design [Sembugamoorthy and Chandrasekaran, 1986], [Sriram and Maher, 1986], [Ulrich, 1987], [Navinchandra and Sycara, 1989], and to reason about new mechanisms by analogy [Falkenhainer, 1986], or from functional descriptions [Doyle, 1989].

1.2. Related Psychological Work

Psychological models have been proposed to describe physical reasoning capabilities in humans, from an experimental framework. For example, psychological models have been developed for physical concept classification [Rosch, 1978], [Murphy and Medin, 1985] and development [Metz, 1985]. The representations that people use to understand the physical environment (their naive mental model) have been studied extensively [Gentner, 1983], [McCloskey, 1983], [Norman, 1983], [Kempton, 1986]. Other models describe performance associated with language comprehension [Bartlett, 1932], [Ausubel, 1960], [Kozminsky, 1977] and problem-solving tasks [Kintsch, 1986, Kintsch, 1988]. Many models have addressed memory access [Thomson and Tulving, 1970], [Tulving, 1972], [Craik and Tulving, 1975], [Glenberg, 1979], retrieval performance [Smith et al., 1978], [Smith, 1979], [Bjork, 1988], [Bjork and Bjork, 1988], and organization [Tulving, 1983]. Experimentation has been studied as a developmental process in learning [Bullock et al., 1982], and [Klahr and Wallace, 1976], and design creativity in children and adults have been studied in [DeBono, 1980], and [Dietterich, 1986], respectively.

1.3. Integration: A Goal of EDISON Project

When the four research areas mentioned above are considered collectively, the representational and processing needs of our research show similarities with, and gaps in, existing research (center of figure 1).

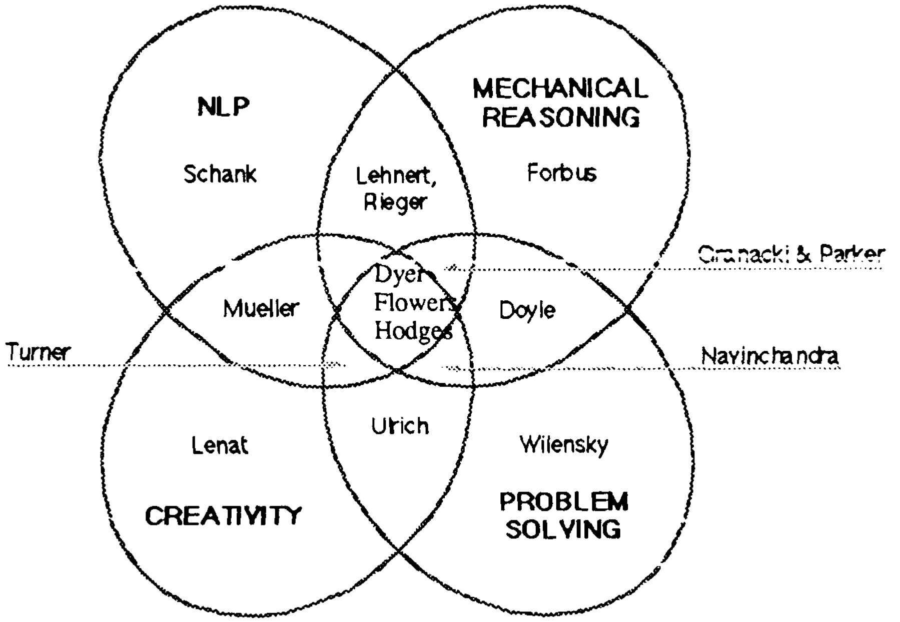


Figure 1: An invention model must combine abilities in each of four research domains: natural language processing, mechanical reasoning, problem-solving, and creativity. Authors of representative research are shown in each area. Our research domain is the intersection of all four research domains.

Critical to the systems in all four areas in figure 1 is the knowledge representation which supports their implementation, along with the processes of knowledge retrieval and application required in each task domain. Each of the above psychological models and computational systems has addressed important issues in their respective task domains. However, no computational (or experimental) model has been developed to address the mechanical invention problem directly. Toward this end we chose to study the domain of simple mechanical devices and how they are comprehended and used in real-life situations by people with little or no technical training (i.e. *naive mechanics*). We believe that this goal can be achieved through an understanding of how everyday objects, such as can-openers, nail-clippers, doors, and mousetraps are represented and reasoned about.

Our objective has been to construct a model for representing and reasoning about mechanical devices, and to illustrate the usefulness of the model with computational experiments in comprehension, simulation, improvisation, and invention.

1.4. Publications Resulting from EDISON Research

During the period of ONR funding, our research has resulted in a number of publications on knowledge representation [Dyer et al., 1987a], [Hodges et al., 1987], understanding mechanical descriptions [Dyer et al., 1987b], object experimentation and mutation [Dyer et al., 1986], situation comprehension [Dyer et al., in press], [Hodges, 1989], and associated memory models [Hodges, 1988], and [Lange et al., 1989]. The most recent publications, [Hodges 1989] and [Lange et al. 1989], are attached as an appendix to this report. Hodges is currently writing his Ph.D. Dissertation. We anticipate that this document will be completed within the next 8 months. Upon completion, it will be delivered to ONR as a report.

2 EDISON ARCHITECTURE

The EDISON system, shown in figure 2, is comprised of eight components:

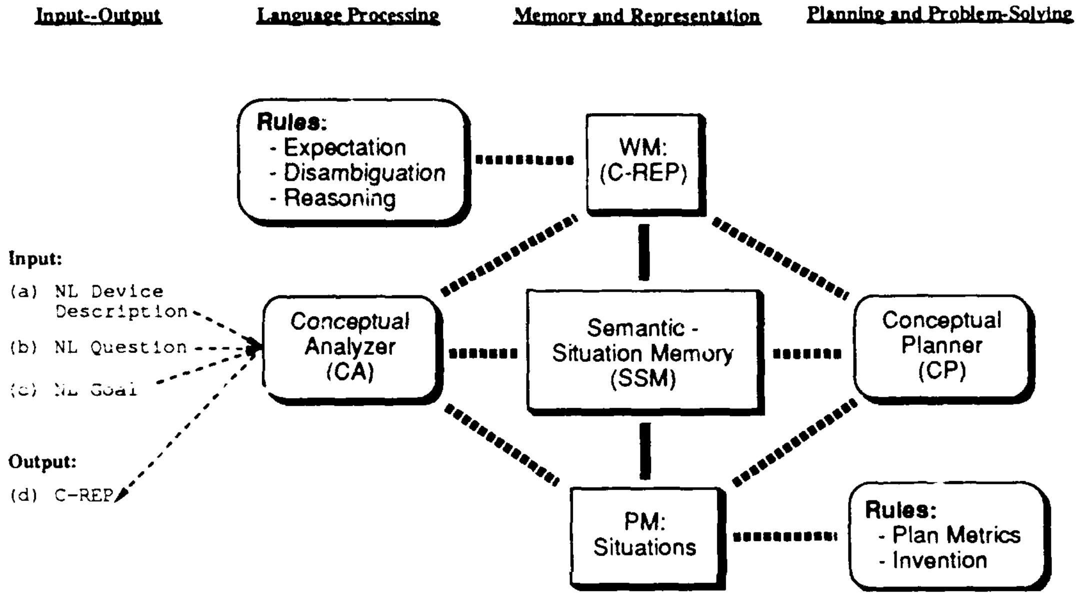


Figure 2: The EDISON process model. Input and output are designated by narrow dotted lines. Controllers are represented by rounded rectangles, and memories are represented by rectangles. Communication between memories is designated by thick, solid lines. Communication between memories and controllers is designated by thick, broken lines. The episodic memory (EPM) is not explicitly illustrated but can be considered similar to WM.

1. *Semantic and Situation Memory (SSM)*: Devices and device-related situations are organized into a memory whereby they can be retrieved and reasoned about. Devices and situations are both frame-like knowledge structures which organize associated causal inferences.
2. *Working Memory (WM)*: During a parse, intermediate conceptual meanings are placed onto a local memory called WM. This memory remains in effect until a sentence is disambiguated.
3. *Conceptual Analyzer (CA)*: Textual input is processed word by word by the conceptual analyzer. Each word is potentially associated with many semantic structures in SSM. These potential structures compete for the word meaning. During parsing the words are presented and disambiguated from the local context. The results are placed onto WM. When a sentence is entirely parsed the conceptual structures remaining on WM are compared to SSM to disambiguate the overall meaning.
4. *Language Processing and Reasoning Rules*: During conceptual analysis rules are needed to comprehend text. Three types, for word-sense disambiguation, clausal reference, and sentential disambiguation, are used.
5. *Episodic Memory (EPM)*: The current conceptualization, regardless of task, is placed onto EPM. This represents the system's mental model of the text or goal.
6. *Conceptual Planner (CP)*: Problem-solving is controlled using an agenda and rules just as the CA.
7. *Planning Memory (PM)*: During problem-solving tasks, situations and planning-related knowledge is retrieved from SSM and placed onto PM, much the same as WM for parsing. This local memory is used as a blackboard for reasoning, and results are placed onto EPM.

8. *Plan and Object Adaptation Rules*: During problem solving, rules are needed to choose between plans, to modify plans or objects, to perform simulations or experiments.

EDISON is designed to function in one of two modes: language comprehension and problem-solving. In language comprehension mode, EDISON receives textual input which is passed to the conceptual analyzer. From there, an initial comparison to SSM is made for word-sense disambiguation and the result is placed onto WM. When the sentence is fully input, the partially-instantiated structures in WM are used as retrieval cues in SSM for sentential disambiguation. The overall result is returned in EPM as a conceptual output. Although figure 2 shows NL question/answering and generation, these are not implemented in the system at present. In problem-solving mode, EDISON receives a conceptual goal to the conceptual planner. The conceptual goal is used as a retrieval cue to SSM, and the situations and objects retrieved are placed onto PM. The CP then coordinates further analysis and retrievals in much the same way as the CA and WM for language. The result is also placed onto EPM.

Language comprehension, experimentation, and improvisation tasks all rely on a conceptual representation and a memory of devices and experiences which can be accessed and modified. The experimentation subsystem recognizes and predicts behavior, or flaws, in device function. It may apply plan modifications to gain new information or remove design flaws. The improvisation subsystem adapts, combines, and mutates device-related situations to produce novel device representations.

2.1 Representing Object Use and Function

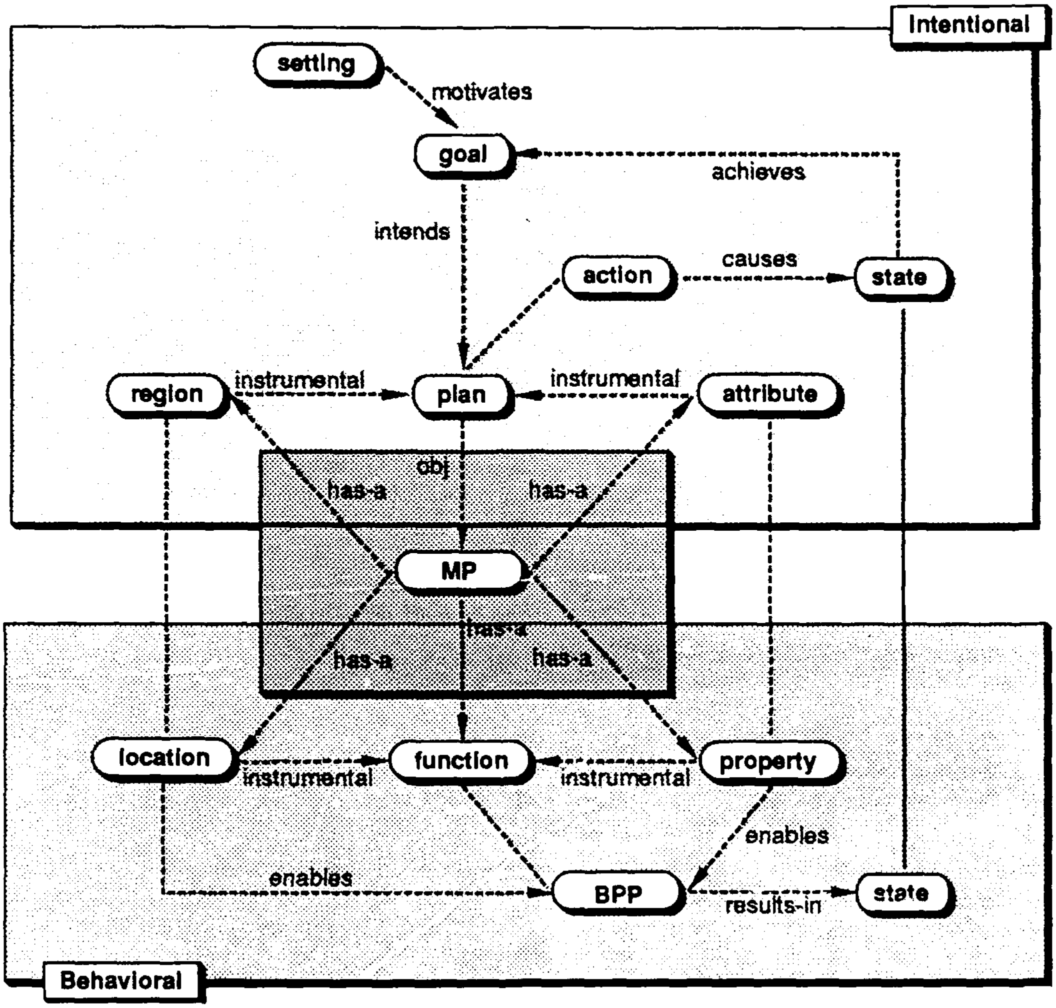
We describe objects at two abstraction levels, behavioral and intentional. Representing object behavior amounts to a description of the causal factors effecting state changes and how they support object function. Representing object use involves a description of the plans in which objects are used, and the goals which they help to achieve.

The representational model is comprised of three components: (1) a behavioral component which describes the interactions of simple objects, (2) a machine component which describes the functional interactions of objects which comprise devices, and (3) a plan component which describes object-use for achieving goals in context. These components are diagramed in figure 3. Intentionally, objects are used in plans to achieve specific goals. Object use is effected through actions/motions, resulting in observable state changes. Behaviorally, objects are perturbed and undergo physical process transformations, resulting in the same states (if the plan is appropriately applied). Plans and actions differ from functions and physical processes in intentionality and variability. A physical (or behavioral) process describes a series of state changes in space and time, and is independent of agency (who is using the object) and context of use.

Figure 3 also illustrates a knowledge dependency graph between representational structures in EDISON. Causality is defined between linked structures. For example, object function is comprised of a sequence of *behavioral process primitives* (BPPs) which describe object behavior. The dotted link between function and BPPs illustrates that there may be more than one pathway described between the two. BPPs are enabled by object physical and relational properties (such as size, location, and material), and result in state changes.

The links illustrated in figure 3 can be followed bidirectionally, and have inverse link names which aren't illustrated in the figure. For example, if a circular (shaped) object *enables* a rolling (Move) BPP, the BPP is *enabled-by* the object shape. In addition to enablement-type links there are disablement links which also are not shown. These will be described later. Finally, two links (initiated-by and terminated-by) describe the states bounding object function. These are not shown in the figure. The indexing adopted in EDISON augments that in [Dyer, 1983].

Figure 3: EDISON knowledge dependency graph. The upper block illustrates the dependencies between object uses (plans) of MPs in intentional knowledge structures (goals and actions). The lower block illustrates the dependencies between MP function and the behavioral process primitives (BPPs) which effect state changes. Machine primitives (MPs) are representational structures for describing device composition, function, and use. Links are shown as unidirectional but are bidirectional. Settings and properties describe environmental and object states, respectively. Dotted lines with arrows show flow of causality. Dotted lines without arrows represent non-direct causality. Solid lines are equivalence links.



2.2. Knowledge Primitives in Mechanical Reasoning

The EDISON model utilizes conceptual primitives to represent the knowledge and reasoning associated with mechanical objects. Primitives represent a categorization level which enables broad concept classification with simple, basic patterns. For this reason the use of primitives 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 composed to describe complex intentional behavior (such as scripts, plans, and goals).

In EDISON we recognize that reasoning about objects occurs at both intentional and behavioral abstraction levels, and that both must be maintained in order to support the reasoning abilities of naive mechanics. To illustrate, when an actor uses a door to enter a room he might turn the doorknob and pull on the door. Turning and pulling are examples of PROPEL actions which resu't in door movement and an open (door) state. To describe object function EDISON must be able to recognize that an actor's actions serve to initiate door functionality, and that the causal relationships which occur internal to the doorknob/door device result in the behavior which the actor observes. It

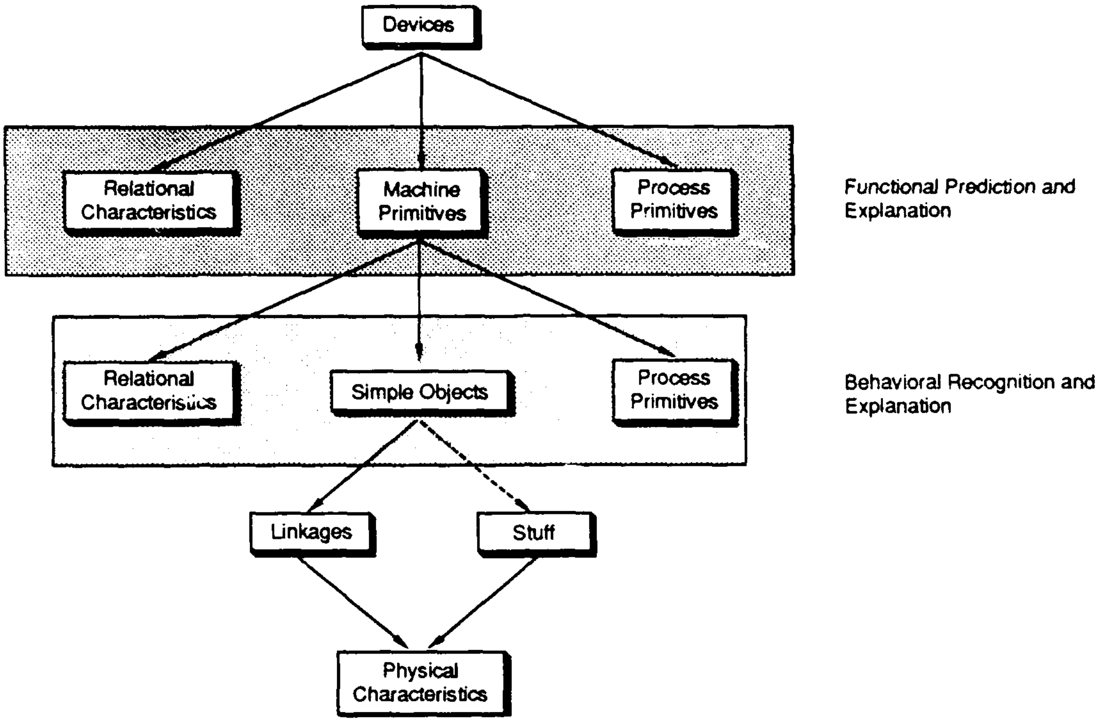

Devices
Relational Characteristics
Machine Primitives
Process Primitives
Functional Prediction and Explanation
Relational Characterisiics
Simple Objects
Process Primitives
Behavioral Recognition and Explanation
Linkages
Stuff
Physical Characteristics

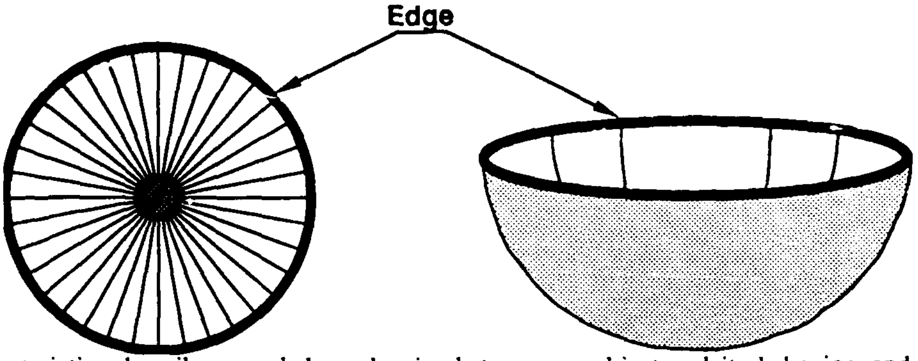
Figure 4: Mechanical decomposition in EDISON. All devices are composed of machine primitives related through process primitives and relational characteristics. This level is used for high-level functional prediction. Machine primitives are decomposed to simple objects and their behavior in process primitives. This level is used for recognition and debugging. Simple objects are described by their physical characteristics.

is simply another level of detail, but the fact that the initial and final states are the same requires that the same overall inferences regarding object use and function be consistent between abstraction levels. Moreover, the fact that the actor turns the doorknob and that the result is an applied force on the doorknob, which initiates doorknob rotation, suggests that causal behavior and inferences be continuous between abstraction levels. Obviously people don't spend their time thinking about doorknob rotation when they want to enter a room. They know how to use a door and they do so. However, when a door is locked or or jammed most people are capable of reasoning and explaining the behavior. Bullock, Gelman, and Baillargeon have discussed the developmental aspects of causal reasoning in [Bullock et al., 1982]. They present a number of experiments which suggest both use and function reasoning capabilities in children and adults.

We introduce primitives which describe simple behavioral interactions, called *behavioral process primitives* (BPPs). BPPs are analogous to actions in that they are causally related to state changes. There are two differences: BPPs describe temporal dependencies, and BPPs are not dependent on agency. We also introduce primitives which describe generalized knowledge and reasoning associated with objects. These are called simple *machine primitives* (MPs), and are analogous to scripts in that they organize BPPs (as plans organize actions) with specific object characteristics. The manner in which object behavior and function are organized to describe complex devices and behavior is depicted in figure 4.

2.3 Causal Interactions Between Simple Objects

In EDISON, all objects decompose to *simple objects* of two types: some amorphous STUFF (e.g. gas, liquid, goo), and LINKAGES. LINKAGES are objects comprised of material capable of transmitting force along [at least] one of its dimensional axes. The behavior of simple objects is affected in three ways: by physical characteristics, by relational characteristics, and by composition. An object's physical characteristics include its material properties, shape, size, and location. Relational characteristics describe topological relationships between objects (e.g. collinear, parallel, offset), and placement of objects with respect to one another. Object composition describes object subcomponents (such as the engine's cooling system, or the belt in a pulley configuration).



Physical characteristics describe causal dependencies between an object and its behavior, and are helpful in recognizing, predicting, and constraining object functionality. For example, a class of generalized locations (called *regions*) is introduced to account for functional object locations. Consider the two circular objects in figure 5.

```
BICYCLE-WHEEL:     BOWL:
(phys-obj BICYCLE-WHEEL                    (phys-obj FRUIT-BOWL
   is-a       WHEEL-AXLE                      is-a       CONTAINER
   has-phys   BW-RIM                          has-phys   FB-RIM
   has-phys   BW-AXLE                         has-phys   FB-BASE
   has-phys   BW-CENTER)                      has-phys   FB-INSIDE)

(region BW-RIM                             (region FB-RIM
   is-a     EDGE                           is-a     EDGE
```

Figure 5: Both wheels and bowls are objects with circular edges. The rim orientation helps to infer the likely function of the object. A vertically-oriented rim is associated with rolling, while a horizontally-oriented rim is associated with containment. Has-phys is a link relating objects and their physical characteristics. The regions axle, rim, center, base, and inside are characteristics. Has-rel is a link relating objects. The orientations vertical and horizontal are relations.

```
   has-rel   VERTICAL)                        has-rel  HORIZONTAL)
```

All circular objects have an *edge* region which defines a circle. If the object is oriented such that the edge is in a vertical plane (e.g. a `BICYCLE-WHEEL`), then we tend to assume that the object is used for rolling. If the edge is oriented in the horizontal plane (e.g. a `FRUIT-BOWL`), then we tend to assume that the object is used for containment. Thus, in this example, by recognizing a single region and its orientation we can make inferences about an object's potential functionality.

2.4 Behavioral Process Primitives - BPPs

Qualitative reasoning systems utilize generalized experiential knowledge of physical behavior to support non-quantitative reasoning about object behavior. Qualitative models describe object behavior as causal dependencies between objects. Any physical interaction which results in object behavioral dependencies is called a behavioral *process*. A behavioral process is a knowledge structure which describes object physical (and relational) characteristics and how they affect time-dependent state changes. Processes are represented according to the Forbus qualitative process (QP) theory. In this model, processes have five roles: objects, physical preconditions, behavioral preconditions, influences, and resulting states [Forbus, 1983].

The EDISON model describes physical interactions in terms of five behavioral process primitives (BPPs): CONSTRAIN, MOVE, TRANSFORM, STORE, and DEFORM. CONSTRAIN and MOVE represent the most simple processes, and the others are combinations of these as shown in table 1. BPPs have three characteristics useful for computational reasoning:

1. BPPs support qualitative inferences and simulation
2. BPP exhibit inheritance.
3. BPPs superpose to describe complex physical behavior.

Qualitative Behavioral Inferences: Each BPP can be recognized from the qualitatively different physical states they effect, as shown in table 1.

BPP	Dependencies	Objects	Causal Relationship (results-in)
Consurain	none	2	physical constraint
Move	Constrain	1	physical position
Transform	Constrain, Move	2	applied force or motion dimension, direction, or magnitude
Store	Transform	1	elastic size, shape, and internal energy
Deform	Store	2	plastic size and shape

Table 1: Behavioral process primitives (BPPs). BPPs differ primarily in their resulting states and process dependencies. CONSTRAINs, TRANSFORMs, and DEFORMs always involve two objects, MOVEs and STOREs involve one.

If an effect is observed then the process can also be inferred. Likewise, if the process can be recognized from the physical attributes, then the effect can be predicted. For example, suppose that an object is moving along and suddenly stops. Given the new constraint state for the object we can infer (using table 1, row 1) that some process has occurred which disables object motion and constrains the object in this dimension and direction.

All CONSTRAIN processes result in object constraints for specific dimensions and directions. Since all BPPs depend on the CONSTRAIN process, each process is effected for some dimension and direction. Dimensionality is applied as force or constraint states. Consider the MOVE process, which describes object motion. MOVE in a dimension and direction is *enabled-by* constraint freedom in that dimension and direction. The MOVE is *initiated-by* an unbalanced force state in the same dimension and direction. The MOVE *results-in* a position state change, also in that dimension and direction .

Process Generality: The organization of BPPs is hierarchical, so that specializing any process role constrains the inferences that can be made regarding the process and the affected objects. We will illustrate this relationship with the CONSTRAIN processes because they affect all other processes. There are three CONSTRAIN types in EDISON: contact, interfere, and connect. Contact is the least-defined CONSTRAIN, is effected by two objects sharing the same position, and results in constraint states for each object in opposite directions along the contact dimension. Interfere is a specialization ot contact where the resulting constraint state effects the entire dimension. Connect is a specialization of interfere where a fastener is used to effect the interference. Table 2 illustrates a few CONSTRAIN specializations.

Name	Dimension	Type	Relational	Fastener
Support	Vertical	Contact	above	none
Hang	Vertical	Contact	below	none
Fit	?dim	Interfere	none	none
Tie	?dim	Connect	none	Rope
Bolt	?dim	Connect	none	Bolt/Nut

Table 2: CONSTRAIN specializations. Support and Hang are contact processes which differ in the object's center of gravity is above or below the other, respectively. Fit is a contact in which one object is constrained by another along an entire dimension (?dim) but they are the only objects. Tie and Bolt are connect processes because they require a fastener to effect the constraint states.

Table 2 describes patterns that can be used to reason about objects in contact. If a book is on a table, then we can infer that it is supported by the table. If a picture hangs from a wall we infer that the bulk of its weight is below the nail. By specializing these processes the inferences which can be made regarding object participation in other processes is further defined. At the same time, the object inherits inferences from the generalized process.

Process Assumptions: In EDISON we define two assumptions which affect process reasoning: (1) enabled processes continue until otherwise acted upon, and (2) objects are free to move in a dimension and direction unless otherwise constrained. These assumptions, and the composition of BPPs to describe complex physical behavior are illustrated in figure 6.

Figure 6 shows the CONSTRAIN process constrain:interfere instantiated for a screwdriver prying function on a silver-polish can lid (SP-Can-Lid). The screwdriver-tip size being smaller than the sp-can-slot results in the screwdriver-tip sharing positions with the sp-can-lid-lip, and the screwdriver-body sharing positions with the sp-can-lip. The first process assumption states that while constrain:contact-react and constrain:contact-pivot are enabled each object has a constraint state. The second assumption states that each constraint state disables screwdriver participation in a MOVE process in the dimension and direction of the other object unless the other object also participates in the MOVE process. The effect of each contact in figure 6 is to block screwdriver movement in one direction in its along-width dimension.

The second process assumption states that objects are free to participate in MOVE processes unless otherwise CONSTRAINed. Because the objects are only in contact, when either constrain:contact process is disabled the object can again participate in a MOVE process, in the specified dimension and direction. Thus, when the SP-Can-Lid is removed from the SP-Can, the Screwdriver will again be able to move in its along-height dimension, positive direction.

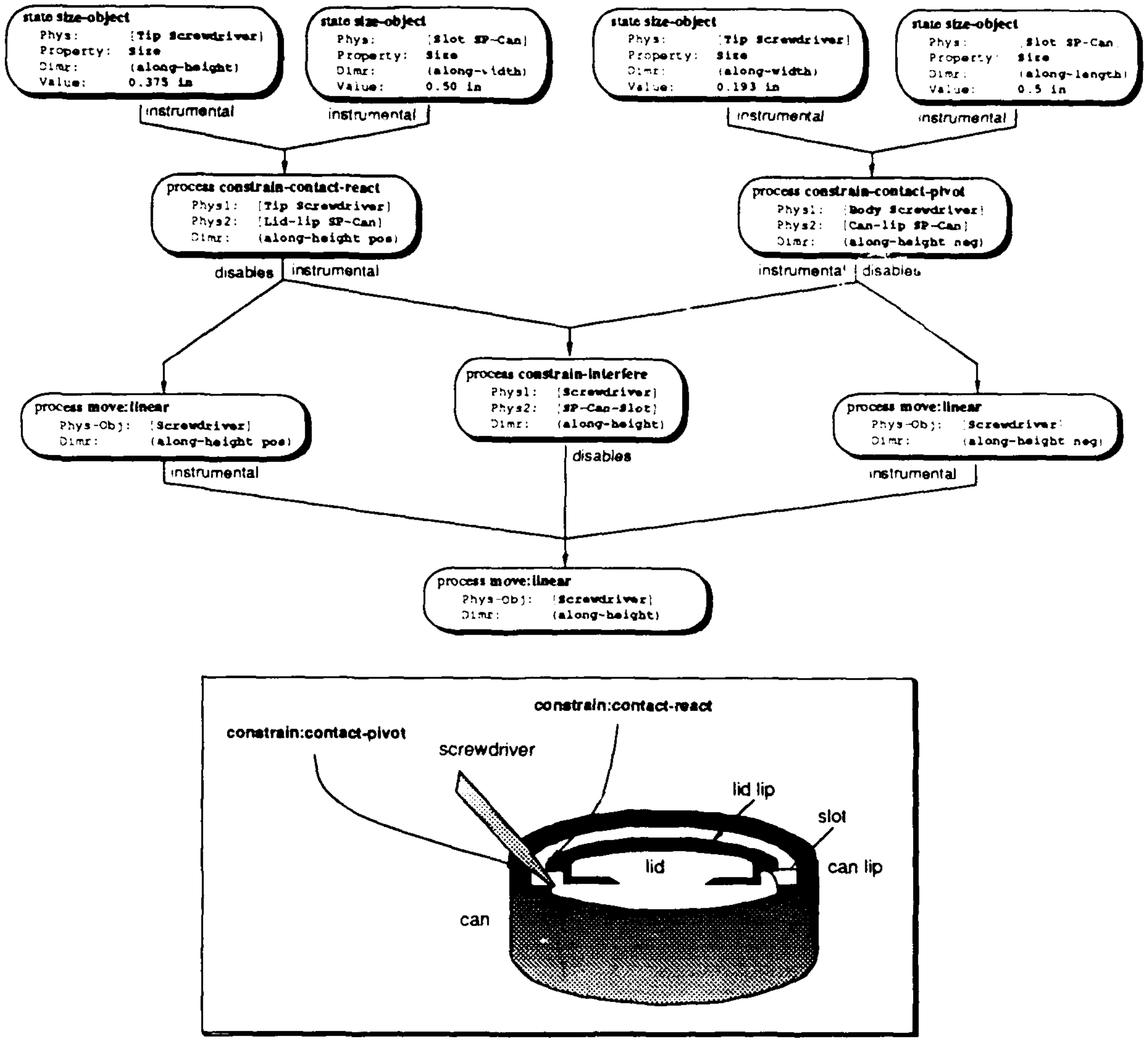


Figure 6: Contact and Interference CONSTRAIN processes instantiated for silver-polish type can (sp-can-lid) prying with a screwdriver. The upper diagram illustrates the causal affect of screwdriver and SP-Can physical characteristics on the contact processes required for prying and SP-Can-Lid motion. The lower diagram pictorially illustrates the relationships in the upper diagram.

Behavior Superposition: When objects interact their behaviors accumulate. In the example above, the processes `constrain:contact-react` and `constrain:contact-pivot` superpose. Each contact process in figure 6 affects the `screwdriver` in its `along-width` dimens.‿n, near the same location and in opposite directions. The effect is to eliminate the possibility of screwdriver motion along the entire dimension *at that location and time*.

2.5. Object Function: Causally-Sequenced BPPs

Complex object behavior can be described by multiple BPPs linked in causal sequences. We call such sequences object *functions*. In addition to process enablements, each function is *initiated-by*, and *terminated-by*, physical states. These states are often boundary values. Consider the *opening* [function] of a can. The function is initiated by the force state resulting from an agent pulling, or pushing, on some PRY-OBJECT. The function is enabled by the 'closed' can state and, regardless of how 'open' the user wants the can, will terminate when the can-lid is fully removed. Using bounding states aids in predicting object behavior, as well as explaining when object function has backfired. These concepts are illustrated in figure 7.

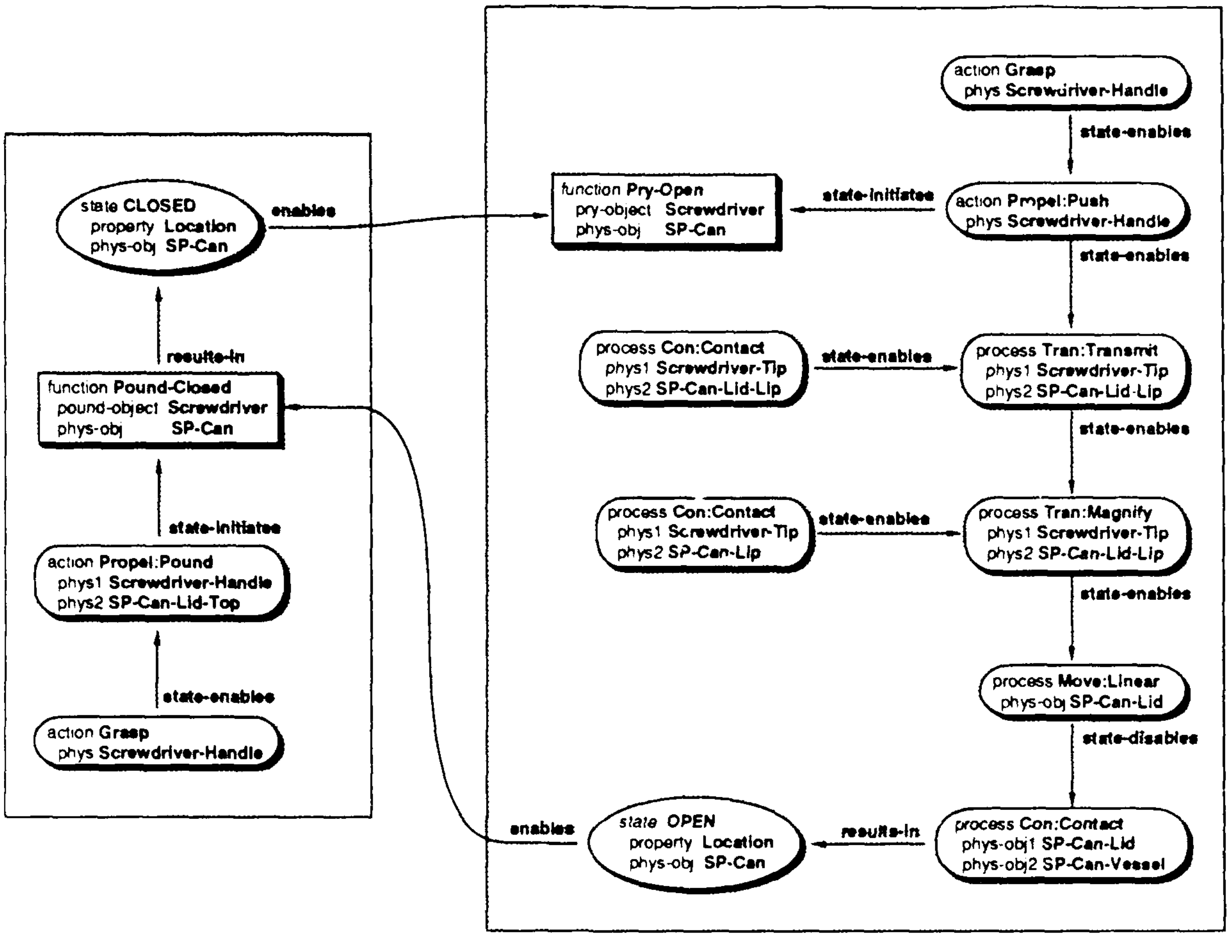


Figure 7: Representing screwdriver Pry-Object functions: the left side shows the high-level (I/O and enablement only) dependencies for the *closing* function; the right side also shows the low-level (process) for the *opening* function. Both functions are shown with variables bound to screwdriver and silver polish can parts and regions. Process dimensions are not shown, and function terminating states are not shown indexed to their functions. The state-enables link is a shorthand for (results-in STATE enables).

The *Pry-Open* and *Pound-Closed* screwdriver functions shown in figure 7 describe different ways that processes are reasoned about in EDISON. The *Pound-Closed* function shown on the left details only the functional enablements, initiating and terminating states. This amounts to the OPEN SP-Can state, the problem-solver's placement of the SP-Can-Lid onto the SP-Can-Vessel (not shown), the action propeling the Screwdriver-Handle, and the CLOSED SP-Can state. This reasoning level is generally used when predicting behavior of recognized or known objects. The *Pry-Open* function shown on the right side of figure 7 describes the underlying process sequence comprised of Con:Contact, Tran:Transmit, Con:Contact, Tran:Magnify, Move:Linear, and Con:Contact BPPs. The process level description supports more detailed reasoning and prediction about the Screwdriver as a PRY-OBJECT and SP-Can behavior. Decomposing screwdriver function representation to the process level supports four deep reasoning requirements:

1. Infering and predicting object behavior
2. Explaining object *mal*function
3. Reasoning about simple temporal relationships
4. Limiting, or bounding, states to object function

1. *Inferring Object Behavior*: BPP and functional level representations are used in EDISON to reason about objects from different perspectives. BPPs describe behavior of object regions. EDISON can make predictions of device behavior given only limited knowledge. If an object is known to be participating in a process but the regions aren't specified, then general inferences can be made about the process (enablement, resulting states, physical characteristics). For example, when a screwdriver is mentioned in text we *expect* some reference to the *Pound-Closed* or *Pry-Open* screwdriver functions. Given an event in either function EDISON can *predict* the processes, and resulting states, which are temporally local to the known event. For example, if EDISON knows that the *Pry-Open* function is active and that force has been transmitted from the Screwdriver to the SP-Can, then the resulting *Pry-Open* state is inferred.

When object regions *are* specified EDISON can trace the 'flow' of force and motion from process to process. Given a state in the overall object function, EDISON infers which process segment is active and predicts the expected state sequence Referring to the previous example, if EDISON knows that the *Pry-Open* function is active and that force has been transmitted from the Screwdriver-*Tip* to the SP-Can-Lid-*Lip*, then [at least] partial enablement of Tran:Magnify is inferred, and EDISON predicts the magnified force at SP-Can-Lid-Lip.

2. *Explanations of Object Malfunction*: EDISON can use the same relationships to explain behavior. Consider the situation in figure 7 where Move:Linear is disabled. EDISON is able to explain this unexpected behavior by noticing that its enablements weren't satisfied, and by comparing the current instantiations in Tran:Magnify to those in a successful situation.

3. *Temporal Relationships*: There are two temporal relation types illustrated in the *Pry-Open* process sequence: sequential and parallel. Often processes result in states which themselves enable other processes. This is *sequential* behavior and can be reasoned about by tracing the causal links between the processes. For example, Con:Contact between the Screwdriver-Tip and SP-Can-Lip enables the process Tran:Transmit between same. At the same time, Con:Contact between the Screwdriver-Body and SP-Can-Lid-Lip enables the process Tran:Transmit between same. Because the two Con:Contact processes are enabled at the same time this second Tran:Transmit becomes a Tran:Magnify. This second relationship is a *parallel* one, because it describes relationships which occur at the same time. During process recognition EDISON uses rules to specialize a process as much as possible, so first the transmit is recognized and then the context for magnify is checked and the magnify is recognized. I will address these rules when the the process of BPP recognition is introduced.

4. *Bounding Object Function*: The information represented in figure 7 enables EDISON to recognize motion of the SP-Can-Lid away from the SP-Can-Vessel as part of an *opening* plan, to associate opening with the *Pry-Open* function, and to predict that the can will be opened. The 'open' and 'closed' states bound the function and are used by EDISON to make inferences about planning. For example, if a *Open-Can* plan is executed and the open state is not realized, then a planner sees this as a planning failure. EDISON recognizes the initiated Pry-Object function but doesn't recognize the terminating 'open' state. At this point the function can be traced at the process level to explain the plan failure and to debug the plan.

2.6. Machine Primitives - MPs

In physical mechanics two mechanisms describe all purely mechanical behavior, the lever and the inclined-plane [Alonso and Finn, 1970]. The U.S. Navy extends this to a set of six commonly used devices; levers, wheels, gears, pulleys, inclined-planes, and screws. In EDISON every mechanical device can be decomposed to combinations of eleven *machine primitives* (MPs): LINKAGES, LEVERS, WHEEL-AXLES, GEARS, PULLEYS, BEARINGS, PLANES, BLADES, SCREWS, SPRINGS, and CONTAINERS. Each MP represents a specific configuration (object composition, physical characteristics, relational characteristics) and set of BPPs as illustrated in figure 4. Each MP is also minimally a LINKAGE object, so the function of each MP minimally describes the TRANSFORM (transmit) BPP. The physical relationships between MPs is shown on table 3. Machine primitives have three characteristics which affect their general use:

1. MPs exhibit inheritance.
2. MPs can be combined to describe more complex devices.
3. MP functions superpose.

Machines are Hierarchically Ordered: The definition of MPs provides a framework for reasoning about objects and devices which utilizes a common representational structure and set of inferences. LINKAGEs (for example clubs, digging sticks, pokers and the like [Oswalt, 1976]) are the most primitive machines. The function of LINKAGEs is to transmit or translate forces and velocities. All MPs derive from LINKAGEs (table 3).

MP	Type	Physical Characteristics	Regions
Linkage	Linkage	Material: Elastic in dimension	Appl, React
Spring	Linkage	Material: Elastic in dimension	Appl, React
Container	Linkage	Region: Inner/Outer Surfaces	Appl, React
Lever	Linkage	Region: Pivot	Appl, Pivot, React
Wheel-Axle	Lever	Shape: Circular	Axle, Body, Edge
Bearing	Wheel-Axle	Shape: Spherical or Cylindrical	Center, Surface
Gear	Wheel-Axle	Region: Tooth, Composition: Chain	Axle, Body, Tooth
Pulley	Wheel-Axle	Region: Slot, Composition: Rope or Belt	Axle, Body, Slot
Plane	Linkage	Shape: Wedge	Appl, Angle, Edge
Blade	Plane	Attribute: Sharp	Appl, Angle, Edge
Screw	Plane, Wheel-Axle	Shape: Cylindrical or Conical	Thread, Center

Table 3: Machine Primitive (MP) classification. All machines arise from a Linkage object, and branch into Lever, Plane, Spring, or Container specializations.

For example, from table 3 LEVERs are defined as LINKAGEs with a pivot. The fundamental difference is that LEVER function involves force or speed *magnification* instead of transmission or translation. Thus LEVERs and their derivatives affect the mechanism of *mechanical advantage* with the BPP TRANSFORM:MAGNIFY. Consider a stick. A stick can be used as a LINKAGE MP (a POKE-OBJECT) to push objects at some distance. By introducing a fulcrum the same stick can be used as a LEVER MP (a PRY-OBJECT) to pry objects apart. The type of input is changed and the the resulting state is qualitatively different. Consider now a screwdriver. A screwdriver is a WHEEL-AXLE MP because it transforms force (and motion) from one radius (the screwdriver handle) to another (the screwdriver tip), thereby magnifying the applied force. The use of the screwdriver in this capacity, as a SCREW-OBJECT, can be inferred from the associated function's resulting state. The WHEEL-AXLE is a derivative of the LEVER MP, so the screwdriver can also be used as a PRY-OBJECT. Likewise the screwdriver can be used as a POKE-OBJECT. Regardless of its use, the inferences associated with LINKAGE MPs can be made of the screwdriver used as a WHEEL-AXLE because they are hierarchically defined.

Different objects can instantiate MPs. The result is that similar inferences can be made regarding their behavior. For example, consider the screwdriver and claw-hammer depicted in figure 8 as instantiations of the LEVER MP as used in the *Pry-Object* function. Each MP is described by regions which define generalized I/O and connection locations. Since LINKAGEs are used for transmitting and translating force, and motion, they are described by two regions; an application point (or appl) and a reaction point (react). The LEVER MP specializes the LINKAGE definition by adding a pivot region, so every LEVER-OBJECT will have three regions (appl, pivot, and react). The middle box in figure 8 shows how screwdriver and claw-hammer regions (upper left and right boxes, respectively) instantiate LEVER-OBJECT regions.

The lower box in figure 8 shows the BPP sequence for a generic PRY-OBJECT function. Using these representation structures and relations, inferences about screwdriver and claw-hammer use as PRY-OBJECTS can be made. For example, the claw-hammer may not satisfy the `Constrain:fit` requirement for prying a paint-can lid whereas a screwdriver would. Another point is that there is no reason why screwdriver and claw-hammer regions cannot be applied as LEVER-OBJECT regions in different ways.

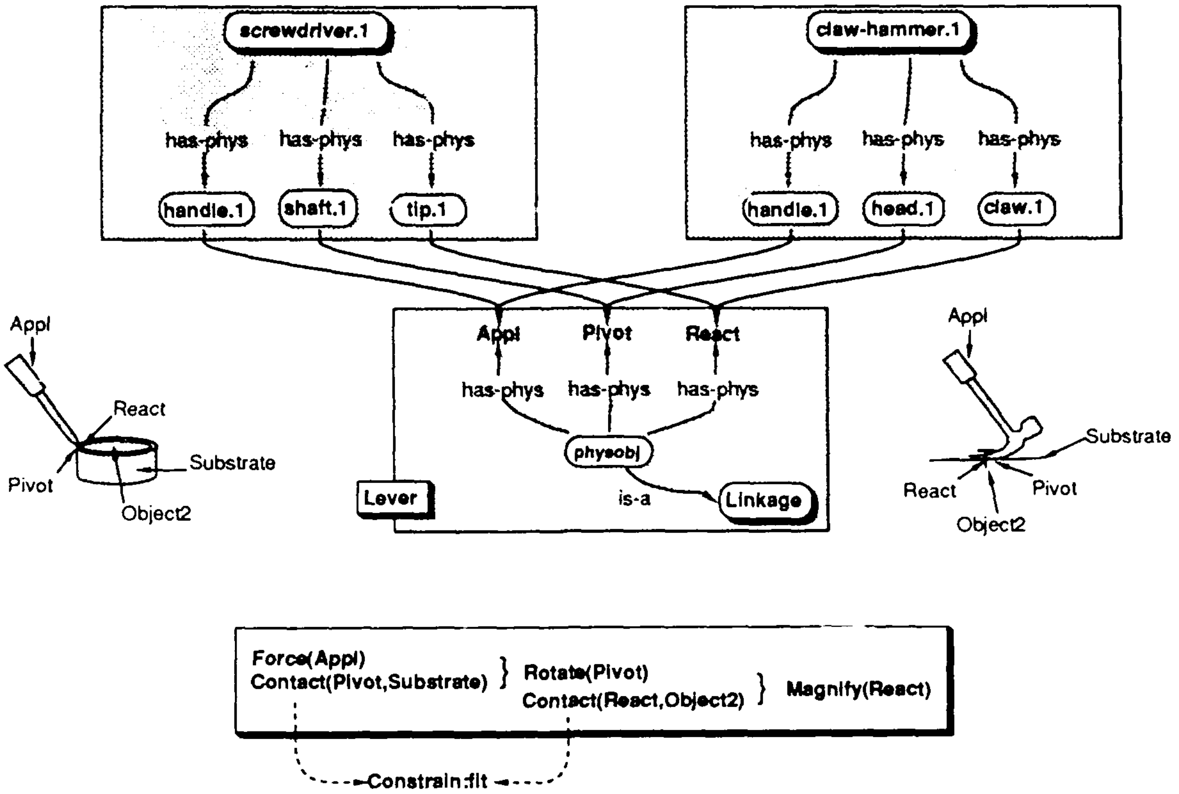


Figure 8: A screwdriver and a claw-hammer are both instances of the machine primitive. LEVER. The representation requires three regions: an appl, or application point, a pivot, and a react, or reaction point. Different regions on the screwdriver and hammer are instantiated as the LEVER regions. Note that these could be instantiated in different ways. The process sequence describing the LEVER function associated with TRANSFORM:MAGNIFY is also illustrated.

Machines are Composable: Most of the objects we use on an everyday basis represent MP compositions. In reasoning about why a composite device works there is an implicit understanding of the objects which comprise the device. Consider the configuration of MPs in a common household device, the CRANK-CAN-OPENER (figure 9). The CRANK-CAN-OPENER (CCO) is comprised of 7 (4 different) primitives: two LEVERs, two GEARs, two WHEEL-AXLEs, and a BLADE. EDISON is able to describe, simulate, and use each unique machine separately.

CRANK-CAN-OPENER

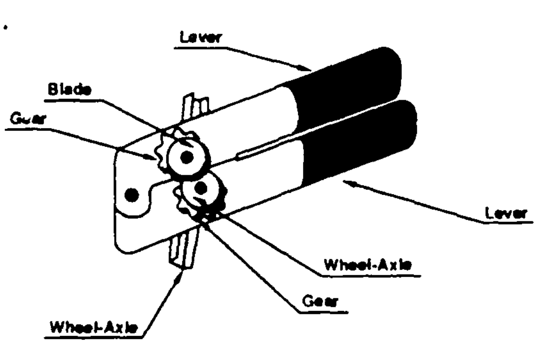


Figure 9: Machine primitives comprise more complex devices as in the CRANK-CAN-OPENER. As unique machines each component can be reasoned about independently.

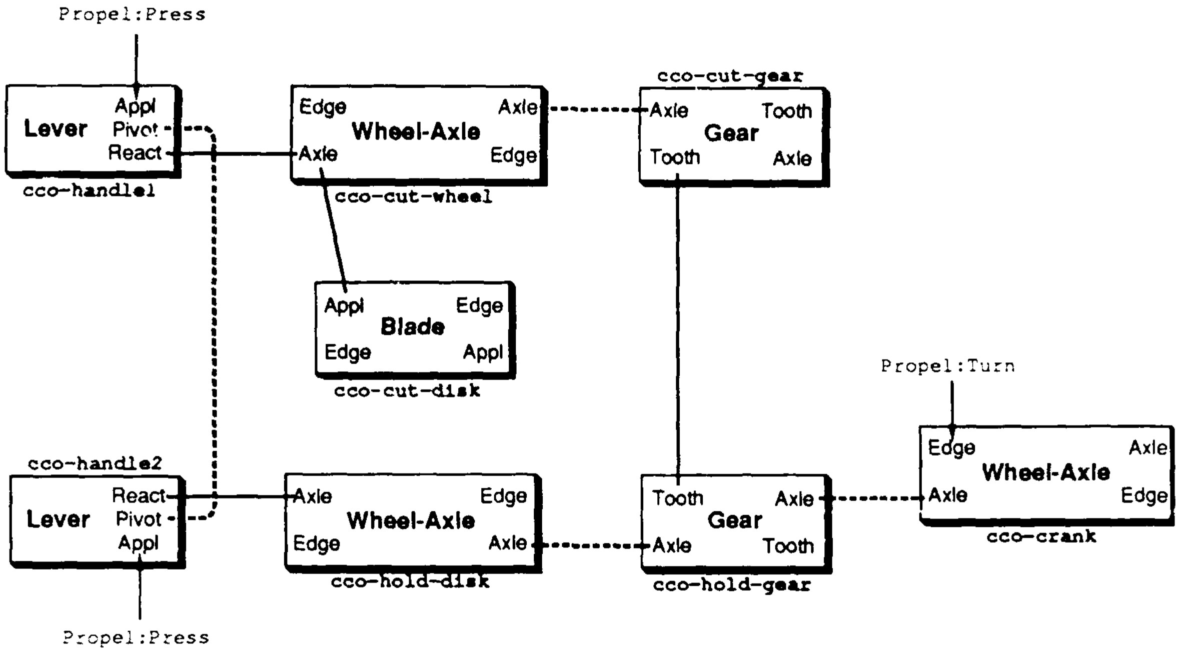


Figure 10: The CRANK-CAN-OPENER as a composition of machines. Flow of force and motion occurs at regional locations. Thin arrow'ed lines represent input to the device, while solid lines represent connectivity. Dotted lines represent shared roles across object boundaries. The actions illustrated represent normal application of CCO can-opening functionality.

The representational composition of the MPs which describe the CCO is shown in figure 10. The machines are shown with their regional roles. Given instantiations for these roles each MP function can be instantiated and reasoned about. Figure 10 describes functional connectivity between CCO MPs. This representation leaves the interpretation of device function wide open. For example, the three actions associated with the (nominal) *Cut-Open* function, Propel:Press at CCO-Handle1 and CCO-Handle2, and Propel:Turn at CCO-Crank, are shown. No mention is made of a can or when the different actions are applied. This information is part of the *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 for the other MPs in the CCO. This raises a point. How do the different MP functions compose?

Figure 10 illustrates two points associated with MP connection. First, we depict the sharing of regions between two WHEEL-AXLE configurations. The MPs associated with the CCO-CRANK, CCO-HOLD-GEAR, and CCO-HOLD-DISK all share the same axle region. In this device (CCO) the axle region is an autonomous object so the MPs, themselves, are part compositions. Thus any force or motion which affects one MP simultaneously affects all three. In the figure this is represented as a dotted line between axles. The same is true for the pivot region of the cco-handles, and for the axle region of the CCO-CUTTER-GEAR and CCO-CUTTER-DISK. The other point is that MP connection is loosely defined so that a device can be visualized by moving MPs around. As long as the MP regions make functional sense an object can be described.

Objects are often designed for a specific function. The physical characteristics which are so-tailored can be used to infer functional-specificity (and constrain other interpretations) regardless of other potential functions the object might have. For example, the CCO-CUTTER-DISK is designed for cutting even though it is a WHEEL-AXLE MP, figure 11.

CCO-CUTTER-DISK:	**CCO-HANDLE1:**
(phys-obj CCO-CUTTER-DISK is-a WHEEL-AXLE part-of CRANK-CAN-OPENER has-phys CCO-CUTTER-AXLE has-phys CCO-CUTTER-CENTER has-phys CCO-CUTTER-EDGE has-fun CCO-CUTTER-DISK-CUT-FUN)	(phys-obj CCO-HANDLE1 is-a LINKAGE part-of CRANK-CAN-OPENER has-phys CCO-HANDLE1-END has-phys CCO-HANDLE1-HOLE)
(region CCO-CUTTER-EDGE is-a EDGE has-attribute ?SHARP)	(region CCO-HANDLE1-END is-a END has-attribute BLUNT)

Figure 11: EDISON Representations for the CCO-CUTTER-DISK and CCO-HANDLE1 objects in CRANK-CAN-OPENER. These representations illustrate regional characteristics and their functional specificity.

Figure 11 shows the CCO-CUTTER-DISK edge (the WHEEL-AXLE region) being *sharp*. When a naive mechanic sees a sharp object they infer that the object is used for cutting. In EDISON a regional attribute overrides other physical characteristics. Despite the fact that the CCO-CUTTER-DISK is circular and has an axle, EDISON first views it as a BLADE MP. Similarly the CCO-HANDLE1, CCO-HANDLE2, and CCO-CRANK all have *blunt* ends. This makes them particularly useful as handles but doesn't obviate their use as LEVERs or WHEEL-AXLEs.

Devices can be described as compositions of MPs, but how are MP configurations interpreted? Are there an infinite number of device combinations? Consider the devices that the CRANK-CAN-OPENER is used to effect. Can their overall function be described as that of a machine primitive? When the CRANK-CAN-OPENER is used to crack nuts, or to pound nails, we can view it as different types of generalized LEVERs (different orientations of appl, pivot, and react regions).

Now consider a clockwork, such as that described in [Nielsen, 1989]. If we try to reason about the clockwork as a sequence of unique states we see that even the most simple child's (transparent) clock is an intractable reasoning task. But the clockwork can be considered a generalized GEAR (a GEAR composition, like a gearbox) which takes some rotational motion and produces some [other] rotational motion. By considering the clockwork as a composite GEAR the computational (cognitive) effort required to understand what it does, and how it works, is greatly reduced. At the MP reasoning level EDISON considers each GEAR pair in the clockwork. EDISON makes inferences about the direction and magnitude of forces and motions transmitted between the gears by considering the input force direction and the relative number of teeth. If there are 10 gears in the clockwork, then EDISON considers 9 gear pairs and makes 18 inferences (direction and magnitude for each pair). Thus EDISON can only determine which direction a clock hand will turn, and its relative speed with respect to the any particular gear. We believe this is ample for the reasoning that naive mechanics appear capable of. When EDISON considers the clockwork as a generalized Gear only two inferences are made: that the device can change the direction and magnitude of rotationally applied force and motion.

Machine Functions Superpose: When introducing BPPs we described behavioral superposition as an additive effect of one object's behavior upon another. As unique objects each MP can be connected to another object and the sequenced process-level behaviors superpose. For example, no matter how many objects we connect together in some dimension they must all move as one object. Similarly, the functions of objects superpose when MPs are connected. If a stick is used as a POKE-OBJECT and a saw is used as a CUT-OBJECT, the two objects can be connected to cut at an extended distance (e.g. tree branches). Whatever forces are applied to the stick appl region are transmitted to the saw and effected at its cutting edge. Similarly, whatever movement the stick makes the saw will see exaggerated due to stick length.

Consider the CRANK-CAN-OPENER illustrated in figure 9. CCO-HANDLE1 and CCO-HANDLE2 are riveted together at CCO-PIVOT. This combination results in two single levers or one combined lever. When the CCO-HANDLE2 is rotated about the rivet the MPs which are connected to it also rotate about the pivot. The magnified force which the CCO-HANDLE effects as a LEVER-OBJECT affects the functions of the CCO-CRANK, CCO-HOLD-GEAR, and CCO-HOLD-DISK. Functional superposition thus guarantees that when objects are connected their functions will be continuous.

2.7 Object Function and Situational Use

Objects can be used to achieve different states in different situations. For example, a can-opener can be used to crack nuts, or hold papers down. The functions which define object behavior are not dependent on the surrounding context in which an object is applied, but the plans which effect that functionality are. In EDISON we define *object use* as the application of an object function in context. Choosing different object functions according to context thus requires a causal interaction between plans and functions. In real situations object use depends on satisfying two types of constraints: (1) the physical characteristics which support object function, and (2) the contextual constraints which support plan application.

2.7.1 Functional Constraints: Regions and Properties

An object function can be applied to a situation if the physical characteristics (material, regional, etc.) which prescribe its function are satisfied. For example, consider the CRANK-CAN-OPENER. How does EDISON choose this device for cracking nuts? The object chosen must satisfy three functional requirements: it must be constrain the nut from moving, 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 be used as a MP if it has the regions which satisfy the MPs definition. A nutcracker is a LEVER MP, and has appl, pivot, and react regions. The CRANK-CAN-OPENER has LEVER-OBJECT components in CCO-HANDLE1 and CCO-HANDLE2. The CRANK-CAN-OPENER thus satisfies the regional constraints of the nutcracker.

2. *Property Constraints*: An object can be used as a MP if its physical properties support the functional behavior. The breaking strength (material property) of a nutcracker is much higher than that of a nut. To use the CRANK-CAN-OPENER as a nutcracker its breaking strength must be comparable to that of the nutcracker. Since the CRANK-CAN-OPENER is used to open metal cans, and is made of metal itself, as is the nutcracker, the CRANK-CAN-OPENER satisfies the property constraint. A claw hammer can also be used to crack nuts using the same reasoning.

Objects which satisfy the regional and property constraints of the same MP are *functionally equivalent* to the MP. We can say that an object has as many uses as it has functional equivalences with other objects. By decomposing devices to other devices and MPs it is easier to describe functional equivalence without making mechanism (process level) comparisons.

2.7.2 Contextual Constraints: Functional Attributes

An object can also be applied to a situation if those regional properties required by a particular function comply with the constraints set up by the situational context. For example, the claw-hammer described in figure 8 satisfies the functional requirements of a PRY-OBJECT, and can be used for removing nails from wood. But a claw-hammer cannot be used as a PRY-OBJECT for opening a paint or varnish can, because the claw is too wide to fit the slot between the can lid and vessel. The claw cannot simultaneously contact the can-lip and the lid-lip. These contacts are necessary to produce the leverage required to overcome the friction between the lid and can.

Both contextual and functional property constraints on object use are affected by property comparisons. These comparisons affect how an object is used in a situation, as well as how it behaves. For example, when choosing an object for prying we tend to look for long objects, not for objects with a specific length. The length affects object function but the term long is sufficient for plan application. The difference is that contextual comparisons like wide and strong do not have direct (one to one) correspondences with the associated property (size in width, and breaking strength, respectively) values. The comparative terms are useful for planning because they include a wider range of potential objects. But they are not correct in that they cannot effectively be used outside the context in which an object attains such (wide or strong) designation. To continue the PRY-OBJECT example, for digging holes a shovel should have a shaft that is longer than a trowel, or a screwdriver used to pry an paint-can lid. With experience a naive mechanic can make knowledgeable comparisons using relative properties which result in successful object application. However, early on such application suggests planning failures.

In EDISON, situational property comparisons are termed *property attributes* because they constrain object function, and because we attribute property status to them. A property attribute is a comparison (>, =, <) between the physical properties of different objects in a particular context. For example, a silk stocking may be *elastic*.

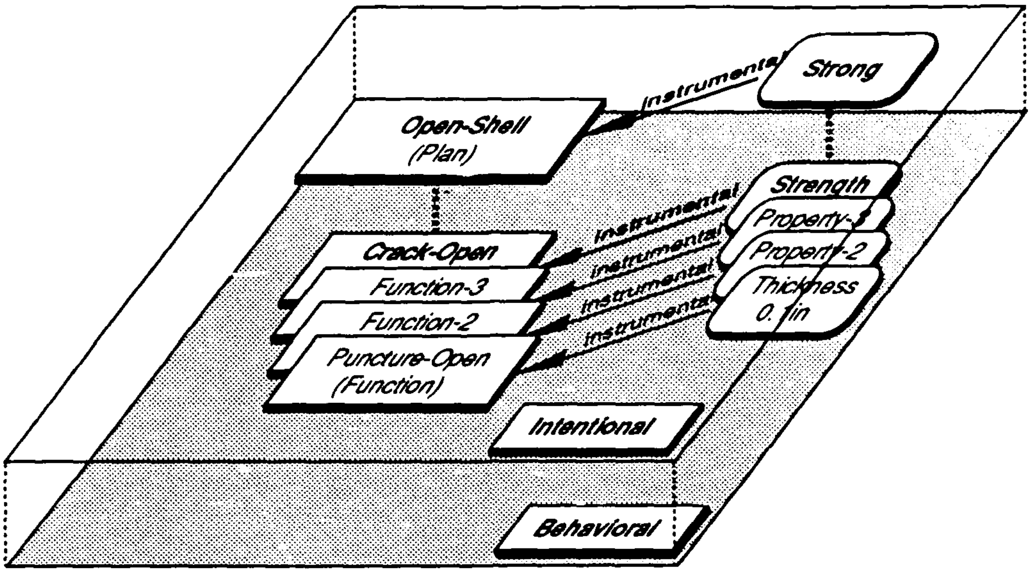


Figure 12: Comparative properties, or attributes, and their relationship with object function and use. The upper level represents the object use (planning) reasoning level. The lower level represents the object function (behavioral) level. Attributes relate use and function through object property values. A CRANK-CAN-OPENER can be used to open nut shells because it is strong compared to the shell. It can also be used to puncture cans open because it is sharp. Note that different MPs in the CCO are applied in each object use.

Physical properties support different object functions, while property attributes support different object uses (figure 12). In this figure object use is illustrated on the intentional (upper) level, and object function is illustrated on the behavioral (lower) level. To use an object in a plan to open nut shells the object must be stronger than the nut's shell. To apply the crack-open object function the object's breaking strength must exceed that of the shell. The same property is being described on both levels but the degree of specificity is different. Each object use is associated with different attributes. To open a soup can the lid must be punctured. This plan requires a sharp object, which references the thickness (property) of the object. The CRANK-CAN-OPENER can be used in either context because it satisfies the attribute and property constraints. Not shown in this figure would be the functionality of a paperweight, the heavy attribute and the associated weight.

In order for the CCO to be used as a nut cracker and a can-opener different MPs are employed. The attribute designation thus indexes plans and functions whether the property values are known or not.

2.7.3 Object Use Situations

Objects have many potential uses, and the more object uses one has experienced (successful or not) the more likely a person has of understanding how an object functions and what contexts it can be applied to. A person who has successfully used an object outside its nominal function is also more likely to try using it in that capacity again. As noted in many experiments, experience affects both interpretation and planning with objects ([Duncker, 1945], [Maier, 1945]). In EDISON a situation is a knowledge structure that represents a problem-solving experience. Situations organize a context, the goals motivated, the planning and actions of actors, the function applied, and the resulting states. Figure 13 illustrates a situation where a screwdriver is applied to the task of opening a varnish-can. The goal, plan, machine, and process level representations are all illustrated in the figure. Screwdriver regions (handle, shaft, and tip) instantiate the Lever MP and satisfy functional constraints. The size of the screwdriver shaft and tip also satisfy the contextual (varnish can slot size) constraints. By satisfying both the functional and contextual constraints the screwdriver is effectively used to pry the can lid.

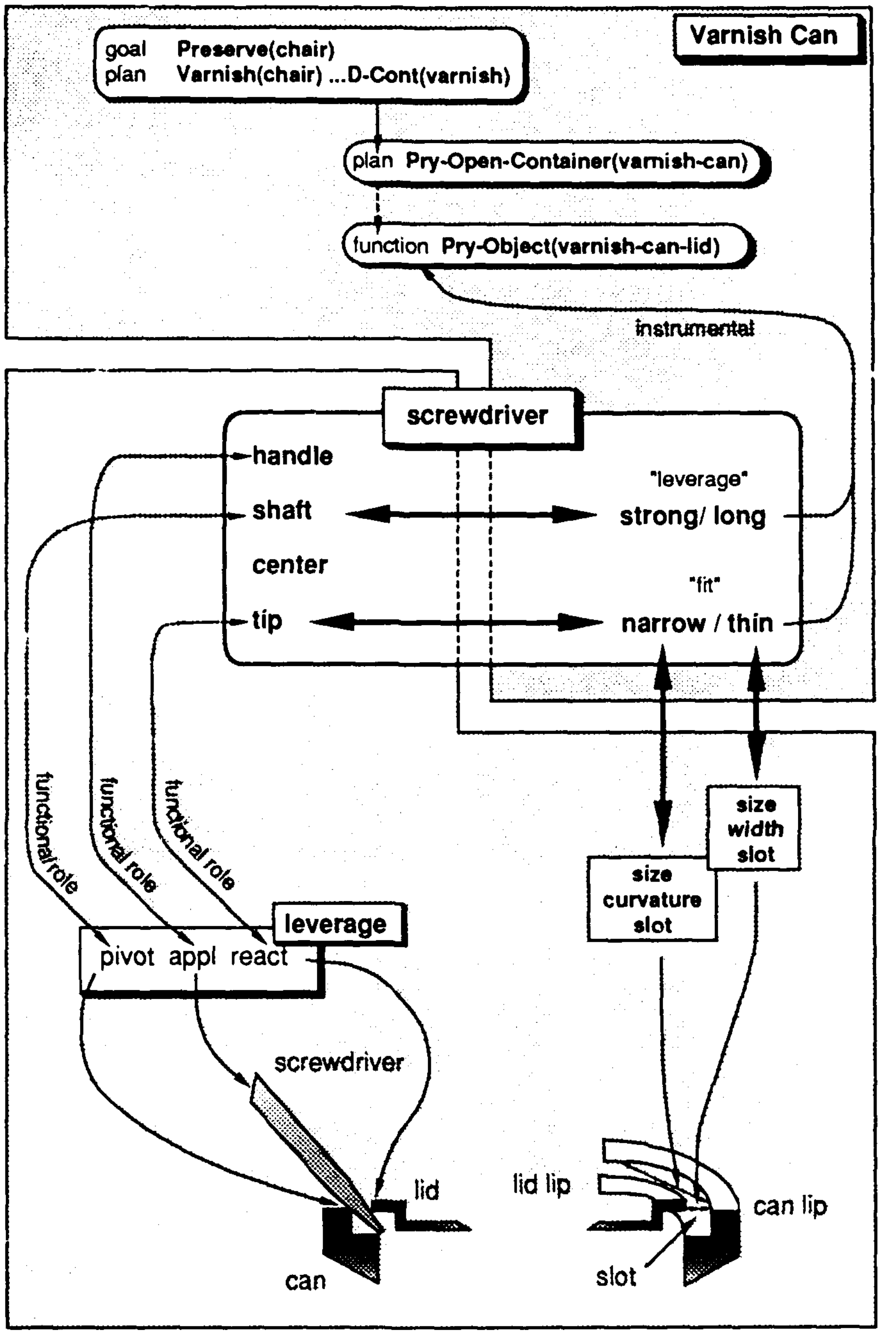


Figure 13: The **Varnish Can** situation illustrates screwdriver use as a PRY-OBJECT in the Pry-Open-Container plan for a varnish can. . The upper block depicts the planning and function in the situation. The middle level illustrates the screwdriver, and the lower level is a pictorial representation describing how the screwdriver satisfies the functional and contextual constraints.

3. PROCESSING TASKS USING MECHANICAL OBJECT REPRESENTATIONS

We have been examining four distinct processing tasks, each designed to test the representational constructs used in EDISON. Currently, there four tasks are being implemented in separate program modules, utilizing different components of the EDISON architecture, each with its own input-output behavior and processing methodologies. For each processing task we are examining a few prototypic examples.

1. *Language Comprehension*: The language comprehension experiment takes as input a behavioral description of toy dart gun function. The model builds a conceptual representation of the objects being described based only on their behavioral interactions. The output is the resulting conceptual representation of the sentential content.
2. *Experimentation and Mutation*: The mutation experiment takes a description of a door and experiments with door functionality by creating different door designs. The output is a conceptual relationship between doorhinges and their causal relationships with door function. The recognition experiment takes examples of objects exhibiting various behavioral interactions. The system uses rules to recognize mechanical interactions, and constructs the associated knowledge structures.
3. *Functional Prediction*: The prediction experiment takes a conceptual representation of a crank-can-opener and some perturbation criteria. The system retrieves similar objects from memory and predicts or explains can-opener function and behavior.
4. *Improvisation*: The situation experiment takes the goals and states describing a problem-solving situation for opening a can of silver polish. The system retrieves the plans and objects associated with experienced situations. These are modified, through comparison and adaptation, to produce an object and plan to resolve the situation.

3.1 Comprehending Device Descriptions from Object Behavior

Device descriptions abound in scientific literature, repair manuals, and technical reports. The difficulties associated with device understanding are pervasive in our literature. The fact that people have general difficulty comprehending physical descriptions, a problem popularized by the Murphy's Law: "If first you don't succeed go read the manual", suggests the significance in comprehending mechanical descriptions.

Understanding an object description is important for describing, remembering, retrieving, and utilizing a design. If the representation and reasoning model work for comprehending device descriptions, there is support for a problem-solving model to support the general invention problem. We have developed a NL comprehension system, Edca (EDISON Conceptual Analyzer), for reading and comprehending mechanical device descriptions. Edca takes a description of object behavior, such as the description of a toy dart gun below, and produces a conceptual representation for the object.

<u>Toy Gun</u>

> "An object is thrust into a barrel, against a spring, compressing the spring until it catches on a trigger. When the trigger is pulled, the spring is released and the object is propelled from the barrel."

When a typical human reads the text of **Toy Gun** they build a mental image of the objects described, and how they interact. The reader infers knowledge missing in the text from their own experiences with similar objects. They can pose and answer questions which pertain to the information conveyed in the description. They can also reason about the events in the description and conjure up explanations for the behavior. Consider the questions posed below.

Q1: What is the initial relationship between the object and barrel?

A1: THE OBJECT IS IN THE BARREL.

Q2: What component catches on the trigger and why?

A2: WHILE THE OBJECT IS MOVING THE SPRING WILL COMPRESS. SINCE THE SPRING STOPS COMPRESSING THE OBJECT MUST CATCH ON THE TRIGGER.

Q3: How is the object propelled from the barrel?

A3: THE COMPRESSED SPRING PUSHES BACK ON THE OBJECT. WHEN THE TRIGGER IS PULLED THE OBJECT CAN MOVE SO THE SPRING PROPELS IT OUT OF THE BARREL.

Q4: What type of motion does the object have?

A4: WHILE THE OBJECT IS IN THE BARREL IT MOVES IN A STRAIGHT LINE. WHEN IT LEAVES THE BARREL IT FLIES, AND EVENTUALLY FALLS.

Questions **Q1-Q4** illustrate both representation-based and processing-based issues associated with comprehending mechanical object descriptions.

Object Behavior Aids Comprehension: Suppose that the reader of **Toy Gun** has no prior experience with guns but some knowledge of objects and their interactions. As the text is read they build a conceptualization of the object's function. This ability requires an ability to reason about object physical and behavioral characteristics. Question **Q1** describes a way to reduce the complexity of reasoning about spatial relationships. If the reader knows about barrels they expect the phrase "pushed into a barrel" to mean that the object is located in the barrel, and that its motion (Q4) is constrained along the barrel length. The need to reason about spatial relationships can thus be simplified by describing object physical characteristics and their affect on object interactions.

Many researchers have approached the reasoning associated with spatially complex objects. DeKleer and Brown [DeKleer and Brown, 1983], Forbus [Forbus, 1983], Faltings [Faltings, 1987], and Nielson [Nielsen, 1989] have all developed physical reasoning systems for describing object kinematics. However, none of these models have been applied to language understanding.

Processing Mechanical Descriptions: The parsing of descriptions such as **Toy Gun** illustrates two issues peculiar to physical texts: object reference, and a causal continuity.

1. *Object Reference*: Naive readers of **Toy Gun** run into a potential problem of keeping the different objects straight from one phrase to the next. They use their knowledge of behavioral interactions between objects to help decide which object is doing what, and how. By remembering how springs work, for example in **Q2** and **Q3**, the reader can infer that the "object" must be catching on the "trigger", rather than the "spring".

2. *Causal Continuity*: A reader of **Toy Gun** mav also experience difficulty inferring continuous and complete causal paths in either sentence. This is because both sentences make assumptions that the reader can infer missing (or low-level) causal paths. The reader must then utilize their understanding of the behavioral relationships between objects to infer, if necessary, the missing pieces. In **Q3** the reader uses knowledge of motion and constraint to infer a missing object motion disablement. This also has the effect of disabling spring compression. The same end can also be attained by understanding that the word "until" (and "when" in the second sentence) implies a causal dependence between object states. Making use of these *causal connectives* both helps to comprehend complex textual descriptions, and to understand the level of understanding the author considered important.

Many researchers have addressed the issue of object reference using semantic knowledge. For example, Dyer used expectations to infer pronouns in story understanding [Dyer, 1983]. Quilici made use of the use of objects in dereferencing data files in the AQUA UNIX advisor [Quilici et al., 1988], while Lebowitz has used references to object composition to dereference device components [Wasserman and Lebowitz, 1983] in patent applications. In EDISON we are trying to utilize behavioral interactions to semantically disambiguate object references in textual descriptions.

Herskovits has done similar work on the comprehension of causal prepositions [Herskovits, 1986]. In EDISON we are trying to understand and taxonomize causal connectives, and to recognize their use in text.

3.2 Experimentation and Invention

Naive invention requires problem-solving. Much of problem-solving involves the retrieval and adaptation of plans and objects to suit a particular situation. Adaptation may require using plans for connecting or separating objects, or modifying them in various ways. Experimentation is a fundamental capability that people use to generate and test hypotheses, thereby obtaining new information that can be applied during problem-solving. Experimentation requires an ability to modify and simulate objects and plans, and to recognize new behavior and form hypotheses regarding that behavior. The less complex the objects and their interactions the better. By limiting the number and complexity of recognized behaviors and their dependencies with object characteristics, it is possible to construct a computational model for mechanical experimentation.

3.2.1 Object Mutation and Hypothesis Generation

The process of experimentation generally involves taking some object, or plan, and modifying it through the choice and application of operator(s). The new object can then be simulated and the behavior compared to the expected behavior of familiar objects. After a few trials a hypothesis may be formed which can be tested by applying it to the operator. The input to an experiment is a conceptual goal, and the output is a conceptual hypothesis. Consider the objects pictured in figure 14 below, and a simple experiment.

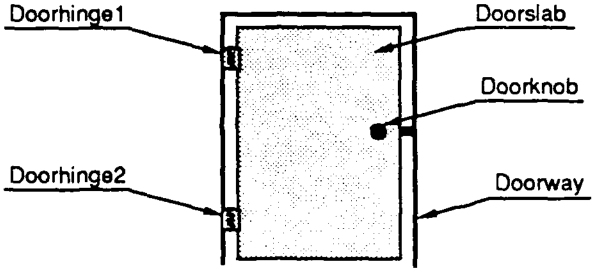


Figure 14: A simple door, with five components (doorslab, doorway, doorhinge1, doorhinge2, and doorknob).

"Experiment with door function by moving Doorhinge1 from one location to another."

Suppose that a child is given a scale model of the door illustrated above. Further suppose that the different parts have velcro on them, so that they can be placed anywhere with respect to one another, and that the hinges allow relative rotation between the plates. If the child is allowed only one operator, MOVE, and a single object, `Doorhinge1`, the child might then begin to move the hinge around and play with the door to see if it still works and how. The questions below illustrate the knowledge and reasoning that the child must have to conceptualize the relationships between doorhinges and door functionality.

Q1: Where can you put Doorhinge1?

A1: ANYWHERE, BUT THE DOOR WON'T DO ANYTHING UNLESS YOU PUT IT ON A SIDE OF THE DOOR.

Q2: Why won't it do anything if you put it on the Doorslab?

A2: BECAUSE BOTH HINGE PIECES WILL BE CONNECTED TO THE DOORSLAB, AND THE DOORSLAB WONT BEND.

Q3: How do you know when a new object is interesting?

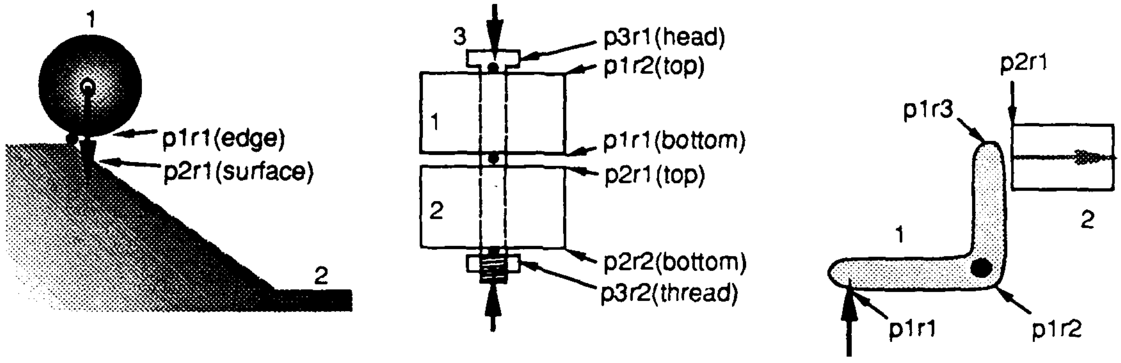
A3: WHEN IT WON'T WORK LIKE THE DOOR.

Q4: When do you decide to stop experimenting?

A4: WHEN I CAN MAKE A DOOR THAT WILL OPEN, OR NOT OPEN, ON PURPOSE.

Answering these questions raises three issues for representing knowledge in computational experimentation: (1) representation depth, (2) hypothesis formulation, and (3) hypothesis evaluation.

Representation Depth: How deeply do naive mechanics reason during experimentation? Given a primitive understanding of motion and constraint the experimenter can determine whether new designs can move, and how. The goal of the experiment is to determine the effect of hinge location on door function (rotation). Simple behavioral knowledge is thus applied in Q1 and Q2 to recognizing whether new doors are capable of rotation. Knowing that the original door can rotate the subject need only determine if hinge movement eliminates this capability. The naive experimenter can thus learn about object function by reasoning about the enablements and resulting states of simple behavioral interactions.



(a) Rolling (b) Bolting (c) Magnifying

Figure 15: Object behavior recognition: (a) a rolling wheel, (b) two bolted blocks, (c) force magnification in a simple lever. Numbers designate objects, and arrows describe applied forces. The p1r1-type notation in each illustration represents a functional location, or region, on an object. These regions can be used to recognize and predict behavior.

Hypothesis Formulation: The child's response to Q3 displays a knowledge that the door experiment is constrained to two possible results; the door will either rotate or it won't. By comparing the different effects of MOVE on door function, the child develops a hypothesis about hinge placement and door rotation; that doorhinges must be on the same side. The subject can thus incrementally build a functional hypothesis by applying operators one at a time.

Hypothesis Evaluation: The child's answer to Q4 illustrates a hypothesis testing phase. By knowing that the door will either rotate or not, the child reasons that generating a door that will work, or not, is sufficient to knowing the relationship between doorhinge and door.

During problem-solving, recognizing physical configurations may require mentally modeling a situation and comparing it to observable data. Simulating the mental model can affect the experimentation process by helping to recognize the behaviors of mutated or perturbed objects. Consider the relationships depicted in figure 15. Given this series of objects engaged in various mechanical and kinematic interactions, naive mechanics can recognize the objects and their associated behavior.

Figure 15 illustrates three simple behaviors: rolling, bolting, and magnifying easily recognized by naive mechanics. Consider now a person troubleshooting an automobile mechanical problem. They may not be able to see these components and their behaviors, and must recognize behavior from what they know, can perceive, or test. They may ask themselves questions regarding the nature of the objects and their behavior, and then make inferences about what is going on. These relationships raise two representation issues for recognizing and predicting object behavior: (1) what behavioral interactions do naive reasoners recognize, and (2) to what extent do naive reasoners use object characteristics to reason about object behavior.

Fundamental Behavioral Interactions

The simple door experiment suggested that motion and motion constraint are fundamental to reasoning about the behavioral interactions of mechanical objects. However, naive reasoners also recognize behavioral interactions which are compositions of motion and constraint. For example, a naive mechanic can recognize that figure 15b represents two bolted objects, and that bolting is a plan for connecting objects. Moreover, they know that once the objects are bolted together the objects will remain so until the bolt rusts or until someone decides to unbolt them. The effect of bolting thus defines a behavioral interaction between the objects which affects their mutual behavior in time. A naive mechanic might also recognize figure 15c as a transformation of force and motion between object 1 and object 2. Transformation involves both motion and contact. The naive mechanic displays a knowledge of object behavior more general than motion and constraint by recognizing transformation instead of the independently recognizing motion and contact. When they see behavior which suggests the transformation of force or motion, they can predict the underlying motions and constraint states. The recognition of these simple behaviors suggests that naive mechanics recognize generic mechanical interactions and classify other interactions as examples of generic behavior. For example, a naive mechanic recognizes the simi:arities and differences between two objects which are bolted, as opposed to being stapled, together. By describing a small set of generic, or fundamental, behavioral interactions the recognition process can be greatly simplified.

We have been studying the composition of simple object behaviors in the hopes of describing the interactions of all simple mechanical objects with a few general behavioral types [Dyer, 1986]. Given a primitive set of behavioral interactions all object behavior can be recognized, composed, and compared more easily.

Object Characteristics and Behavior

A naive mechanic looking at figure 15a recognizes object rolling from the applied and gravitational forces, from the circular shape of object 1, and from its contact with object 2. Were the object pictured square instead of circular no one would expect it to roll. People use physical characteristics such as shape to recognize and discern behavior. They also recognize locations on objects which have meaning to object functional capability, such as the circular edge on a wheel, the threads on a bolt, the pivot location on a lever, or even the edge of a knife. By knowing about generalized locations, and shapes people can reason about how objects interact, and recognize similarities when they are encountered.

DeKleer and Brown have described a "port", which serves as a location where conduits (methods of transporting material or energy from one mechanism to another) connect to an object [DeKleer and Brown, 1983]. Their use of the port idea is similar to our notion of generalized location, or region [Hodges et al., 1987] in that it describes object functional locations.

3.3 Predicting and Explaining Device Function

Device simulation is everpresent in the creative process. Whether a person is recognizing functional similarities between objects, or predicting their behavior, or even experimenting to learn new information about object use and function, the simulation of behavior plays a pivotal role in creativity. The fact that people have limited functional knowledge about primitive mechanisms, and their composition, suggests that simple functional models can be useful in predicting how naive mechanics reason about mechanical objects and their interactions. That people utilize their experience with particular objects as a reminding of what the objects can be used for is suggestive of a model which reasons about objects rather than the mechanisms which objects effect. The use of a simple functional model, one which is based on objects and their similarities, suggests a general reasoning capability with object composition and function. We propose a simple object reasoning model which utilizes the EDISON machine primitives (MPs) to make high-level predictions about the functionality of an object like the CRANK-CAN-OPENER (figure 9).

Most naive reasoners see the CRANK-CAN-OPENER as a composition of bars, wheels, and gears, instead of objects which move, constrain, and transform force and motion in complex patterns. Although they recognize shape, location, and material object characteristics they don't necessarily reason about them unless the object malfunctions. Consider the following questions that can be answered by someone that has knowledge of CRANK-CAN-OPENER function.

Q1: If CCO-HANDLE1 and CCO-HANDLE2 are apart and the CCO-CRANK is turned, will the CCO-CUTTER-DISK rotate?

A1: WHEN CCO-HANDLE1 AND CCO-HANDLE2 ARE APART, THE GEARS CCO-HOLD-GEAR AND CCO-CUTTER-GEAR AREN'T IN CONTACT. WHEN CCO-CRANK IS TURNED THE CCO-HOLD-GEAR WILL ROTATE BUT WON'T TRANSMIT MOTION TO CCO-CUTTER-GEAR.

Q2: If a small object is placed near the CCO-CRANK, between CCO-HANDLE1 and CCO-HANDLE2, and if the CCO-HANDLE1 and CCO-HANDLE2 are brought into contact with the object and then pressed together, how much force will be applied to the object?

A2: IF A SMALL OBJECT IS PLACED NEAR THE CCO-CRANK, BETWEEN CCO-HANDLE1 AND CCO-HANDLE2, THEN CCO-HANDLE1 AND CCO-HANDLE2 WILL ACT AS LEVERS ON THE SMALL OBJECT WHEN CONTACT IS MADE. IF CCO-HANDLE1 AND CCO-HANDLE2 ARE PRESSED TOGETHER, THE AMOUNT OF FORCE TRANSMITTED TO THE SMALL OBJECT WILL BE GREATER THAN THE AMOUNT OF FORCE APPLIED AT CCO-HANDLE1 AND CCO-HANDLE2.

These questions raises an issue regarding the representation of object function for prediction. Do naive mechanics recognize objects which perform specific functions. If so how many such objects do they recognize, and how does temporal reasoning affect prediction of their functionality.

Process Interactions, Mechanisms, and Machines: If a person asked Q1 recognizes the CRANK-CAN-OPENER as a configuration of objects, then they may walk through the application of force, and other behavioral interactions until they realize that the CCO-CUTTER-GEAR and CCO-CUTTER-DISK cannot rotate if there is no transmission of motion from the CCO-HOLD-GEAR. This level of simulation is quite detailed, and requires a consideration of how and where parts are connected. However, most people would recognize that the CRANK-CAN-OPENER is composed of levers, gears, even a blade. They know that the function of gears is to transmit motion. The naive mechanic realizes that gear function is disabled as soon as they recognize that the gears are not in contact when the handles are separated. By recognizing the objects that comprise the CRANK-CAN-OPENER by the functions they have been used to effect, naive reasoners can reason about complicated objects and spatial relationships without the need for explicit simulation at the behavioral level. Being able to answer Q2 suggests that naive reasoners understand both the simple object and the behaviors which its application describes. This reasoning suggests an ability to reason at both representation levels. When possible the recognized object is used to predict macro-behavior. Otherwise behavioral interactions are used to simulate and explain object behavior.

Temporal Continuity: A person using the CRANK-CAN-OPENER to open a soupcan doesn't reason about how long it will take to open the can, or about the periodic behavior of the gears. They know from experience that, once the cut is initiated, and as long as they continue to turn the crank, the cut will eventually return to its starting point. This illustrates one case in which temporal reasoning can be obviated by the type of object used.

Metz's work on the development of physical concepts in children, and in particular with gear trains, has shown that children begin to develop object models (reason about object function instead of object behavior) about the age of 10 [Metz, 1985].

We have been building a model of simple mechanical objects whose functions effect the leverage mechanism. The interactions of these objects are described by our simple behavioral model [Hodges, 1988, 89c]. Our work assumes that naive reasoners have and use object-models during problem-solving, and that their behavioral knowledge is used to explain anomalies rather than to predict object function or use.

3.4 Object Use and Function in Improvisation Situations

When people engage in creative problem-solving, whether constrained as in improvisation or unconstrained as in invention, they utilize an ability to view objects from multiple points of view, with different functionality. This capability requires that they understand the objects they have used in their everyday experience, how those objects have been used, and their similarities to other objects which might be used to achieve the same results. 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 planning and functional level. Regardless of an object's prototypical function, a situation may call for other interpretations of its functional capability. Even though a problem-solver may recognize a functional advantage of one object over another, higher-level goals may cause them to try known objects which are functionally equivalent.

A problem-solving model that focuses on object use to achieve actor goals can be successful. However, such models cannot be extended unless there is an underlying functional model to support the classification of similar objects. We propose a simple problem-solving model based on the EDISON concepts of functional equivalence and the use of property attributes in constraining object choices. Given the ability to view objects as functional equivalents support their use outside known contexts. Given the ability to choose objects based on contextual constraints helps to eliminate irrelevant objects from consideration. Together the application of object-related experience and contextual knowledge can be used for problem resolution in new situations.

Consider the situation described in figure 17.

Prying Knife

A man wants to polish one of his silver candlesticks. To use silver polish he must pry open a can of silver polish in the kitchen. However, the screwdriver is in the garage. He reasons that he can use a screwdriver-like object and decides to try a carving-knife.

Figure 17: A simple improvisation situation. This situation illustrates the use of objects in context and an understanding of objects at the planning, functional, and behavioral reasoning levels.

Prying Knife illustrates an improvisational scenario which might occur to a human being. Resolving problem-solving situations requires access and use of knowledge at different levels of abstraction, and requires us to look at objects in completely different ways. Consider the inferences made in order to answer the following questions regarding **Prying Knife**.

Q1: Why does the man decide to *pry* open the silver polish can?

A1: HE REMEMBERS A TIME OPENING A PAINT CAN BY PRYING, AND THAT BY PRYING THE LID HE WAS ABLE TO OPEN THE CAN WITHOUT DESTROYING THE LID.

Q2: What is a screwdriver-like object?

A2: AN OBJECT THAT CAN BE USED AS A PRY-OBJECT AND CAN FIT THE SLOT IN THE SILVER POLISH CAN.

The questions presented above illustrate two representational issues for creative reasoning about mechanical objects: (1) the role of experience in mechanical problem-solving, and (2) the relationship between object use and object function.

In **Prying Knife** the goal conflict between using the screwdriver and getting wet, and the availability of objects in the kitchen both affect the man's planning and how object functionality is viewed. **Q1** illustrates an interaction between a problem-solver's goals and their retrieved experiences. The application of familiar plans, like using a screwdriver, to resolve the can-opening problem, may be dictated by any contextual element. How the man chooses to view the situation and available objects is determined by his related experiences. He may choose to apply the prying plan by getting the screwdriver, to seek another object to pry open the can, or to cut the can open by applying a different can-opening plan.

When the problem-solver abandons the idea of getting the screwdriver and decides to find an alternative object, he must find an object which is both functionally equivalent to the screwdriver, and fits the contextual constraints. He is reminded of situations where other objects have been used to obtain leverage, and objects that are physically similar. The answer given to **Q2** illustrates both functional and contextual comparisons made in selecting candidate objects. These comparisons require an object representation which makes use of functional similarities and how they are affected by context.

Functional maps between objects has also been the interest of [Falkenhainer et al., 1986]. Their structure-mapping engine (SME) has been applied to comprehending the underlying mechanisms of objects through analogical comparisons with the behavior of known mechanisms.

4 CURRENT STATUS AND FURTHER WORK

Each of the four implementation models has been partially implemented. The representation model presented here has been continually updated along with the development of the respective experiments. At present we can represent, recognize, and simulate the MOVE, CONSTRAIN, and TRANSFORM related BPPs. These have been used in the parsing of **Toy Gun** and the Door experimentation model. The MP reasoning model and situation problem-solving model are only partially implemented. The processing methodologies utilized in recognizing and reasoning about objects in these models are vastly different. When the project was begun symbolic and rule-based methods were being widely used for both representation and reasoning tasks in AI. However, we have been trying to adapt our memory (retrieval and reasoning) model to a localist/connectionist spreading activation computational paradigm [Lange et al. 1989]. This model will address some of the issues associated with scaling the EDISON model to more complex devices and situations.

We intend to extend the the language model to include other demonstration texts, and to complete the development of BPP and MP recognition and inference models to support problem-solving. A major aspect of the work at this point is consolidation and write up, which is the task of Hodges, for a doctoral dissertation. We anticipate that this dissertation will be complete (or nearly complete) with the next 8 months. At that time we will deliver it to ONR as an additional report.

5 LONG-TERM GOALS AND CONCLUSIONS

We are seeking insights in the areas of (a) NL comprehension of device descriptions, (b) symbolic simulations of device behavior and function, (c) device experimentation and improvisation, and (d) device invention. Insights in the processing, memory, and representational aspects of each of these areas can ultimately lead to important applications:

(1) *Device comprehension* can lead to the implementation of (a) tutoring and dialog systems, and (b) automatic conceptual database generation systems. On-line tutors could replace written manuals. Manuals are difficult to read because they require the reader to supply necessary index keys, and manuals are only able to answer anticipated questions. With NL understanding capability, a system could take descriptions from a designer and interpret them (i.e. generate a conceptual representation). This representation would then be indexed into the conceptual database for later use.

(2) *Device Simulation* is useful during mechanical design. One potential application is for design troubleshooting, in which the system could run simulations, notice bugs in device behavior, and suggest possible modifications to objects or plans which would allow the design process to continue.

(3) *Device Experimentation and Improvisation* could allow an intelligent system learn about the behavior of new devices without assistance. Improvisation gives a system the capability of applying old devices in new and unfamiliar environments.

(4) *Device Invention* could be used to rethink design scenarios and support the creation of a *Design Apprentice, i.e.* a design understanding and invention system capable of reasoning and brainstorming about physical devices, their functions and behavior, and capable of communicating that reasoning to the designer.

Automated design and invention relies on representational and processing constructs. Representations must encode a large number of object features (e.g. regions, motions, translations of forces, constraints, connectivity relations, etc.). By building up devices from more primitive machines (e.g. levers), the resulting system can apply inferences in a systemmatic way and complexity is reduced via inheritance and compositionality. The critical cognitive tasks are comprehension, recall, adaptation, simulation, and mutation. Representational constructs can only be validated in terms of how well they support such cognitive tasks.

5 REFERENCES

Alonso, M., and Finn, E.J. (1970): *Physics*. Addison-Wesley, Reading, MA (1970).

Ausubel, D.P. (1960): The Use of Advance Organizers in the Learning and Retention of Meaningful Verbal Material. *Journal of Educational Psychology 51*(1960), 267-272.

Bartlett, F.C. (1932): *Remembering*, Cambridge University Press, Cambridge, UK (1932).

Bjork, R.A. (1988): Retrieval Practice and the Maintenance of Knowledge. In *Practical Aspects of Memory: Current Research and Issues*. John Wiley & Sons, M.M. Gruneberg, P.E. Morris, and Sykes, R.N., pp. 396-401, London, UK, Vol. 1: Memory in everyday life, 1988.

Bjork, E.L., and Bjork, R.A. (1988): On the Adaptive Aspects of Retrieval Failure in Autobiographical Memory. In *Practical Aspects of Memory: Current Research and Issues*. John Wiley & Sons, M.M. Gruneberg, P.E. Morris, , and Sykes, R.N., pp. 283-288, London, UK, Vol. 1: Memory in everyday life, 1988.

Bullock, M., Gelman, R., and Baillargeon, R. (1982): The Development of Causal Reasoning. In *The Developmental Psychology of Time*. Academic Press, Friedman, W.J., pp. 209-254, 1982.

Coyne, R.D., Rosenman, M.A., Radford, A.D., and Gero, J.S. (1987): *A Logic Model of Creativity in Knowled Based CAD*. February 1987, Unpublished research.

Craik, F.I., and Tulving, E. (1975): Depth of Processing and the Retention of Words in Episodic Memory. *Journal of Experimental Psychology: General 104*(1975), 268-294.

DeBono, E. (1980): *Children Solve Problems*, Penguin (1980).

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

Dietterich, T. (1986): Learning at the Knowledge Level. *Machine Learning 1*, 3 (1986), 287-315.

Doyle, R. (1989): *Hypothesizing Device Mechanisms: Opening Up The Black Box*, Ph.D. dissertation, Report TR-1047, Massachussetts Institute of Technology, 1989.

Duncker, K. (1945): On Problem Solving. *Psychological Monographs 58*, 5 (1945).

Dyer, M.G. (1983): *In-Depth Understanding*, MIT Press, Cambridge, Massachusetts (1983).

Dyer, M.G., Flowers, M. and J.B. Hodges. EDISON: An Engineering Design System Operating Naively, *International Journal of Artificial Intelligence in Engineering*, Vol. 1, No. 1, July 1986. (Published by Computational Mechanics and selected for reprinting from *Applications of AI in Engineering Problems Conference*, Two Volumes hardback, published by Springer-Verlag, 1986.)

Dyer, M.G., Flowers, M., and Hodges, J. (1987): Naive Mechanics Comprehension and Invention in EDISON. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Morgan-Kaufman, August 1987, pp. 696-699.

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

Dyer, M., Hodges, J. and Flowers, M. Computer Comprehension of Mechanical Device Descriptions. In J. Gero (ed.) *Knowledge-Based Systems in Engineering and Architecture*. Addison-Wesley (in press).

Falkenhainer, B., Forbus, K., and Gentner, D. (1986): The Structure-Mapping Engine. *Science* (1986), 272-277.

Faltings, B. (1987): Qualitative Kinematics in Mechanisms. In *Tenth International Joint Conference on Artificial Intelligence*, American Association for Artificial Intelligence (AAAI), Morgan Kaufman, 1987, pp. 436-442.

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

Gentner, D. (1983): Mental Models. In *Mental Models*. Lawrence Erlbaum Associates, Gentner, D., and Stevens, A., Hillsdale, N.J., 1983.

Glenberg, A.M. (1979): Component-levels Theory of the Effects of Spacing on Recall and Recognition. *Memory and Cognition 7*(1979), 95-112.

Granacki, J.J., and Parker, A.C. (1986): A Natural Language Interface for Specifying Digital Systems. In *Applications of Artificial Intelligence in Engineering Problems*, Sriram, D., and Adey, R., 1st International Conference, Springer Verlag, Southampton, United Kingdom, April 1986, pp. 215-226.

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

Hayes, P.J. (1978): The Naive Physics Manifesto. In *Expert Systems in the Microelectronic Age*. Edinbugh University Press, 1978.

Herskovits, A. (1986): *Language and Spatial Cognition*, Cambridge University Press, Great Brittain (1986).

Hodges, J., Flowers, M., and Dyer, M.G. (1987): Knowledge Representation for Design Creativity. In *ASME Winter Annual Meeting*. American Society of Mechanical Engineers, Liu, C.R., Requicha, A., and Chandrasekar, S., pp. 81-93, Boston, MA, December 1987.

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.

Kempton, W. (1986): Two Theories of Home Heat Control. *Cognitive Science 10*, 1 (January-March 1986), 75-90.

Kintsch, W. (1986): Learning from Text. *Cognition and Instruction 3*(1986), 87-108.

Kintsch, W. (1988): The role of Knowledge in Discourse Comprehension: A Construction-Integration Model. *Psychological Review 95*, 2 (1988), 163-182.

Klahr, D., and Wallace, J.G. (1976): *Cognitive Development: An Information Processing View*, Lawrence Erlbaum, Hillsdale, NJ (1976).

Kolodner, J.A. (1984): *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*, Lawrence Erlbaum Associates (1984).

Kozminsky, E. (1977): Altering Comprehension: The Effect of Biasing Titles on Text Comprehension. *Memory and Cognition 5*(1977), 482-490.

Kuipers, B. (1986): Qualitative Simulation. *Artificial Intelligence 29*(1986), 289-388.

Lange, T., Hodges, J.B., Fuenmayor, M., and Belyaev, L. (1989): DESCARTES: Development Environment for Simulating Connectionist Architectures. In *Proceedings of the Eleventh Annual Meeting of the Cognitive Science Society*, August 1989, pp. 698-705.

Lebowitz, M. (1980): *Generalization and Memory in an Integrated Understanding System*, Ph.D. dissertation, 186, Yale University, 1980.

Lebowitz, M. (1985): *The Use of Memory in Text Processing*. November 1985, Unpublished research.

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

Lenat, D.B. (1976): *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*, Ph.D. dissertation, STAN-CS-76-570, Stanford University, 1976.

Lenat, D.B. (1983): EURISKO: A Program That Learns New Heuristics and Domain Concepts. *Artificial Intelligence 21*, 1,2 (1983), 61-98.

Maier, N.R.F. (1945): Reasoning in Humans III: The Mechanisms of Equivalent Stimuli of Reasoning. *Journal of Experimental Psychology 35*(1945), 349-360.

McCloskey, M. (1983): *Naive Theories of Motion*, Lawrence Erlbaum: Hillsdale, NJ (1983)pp. , 299-323, Chapter 13.

Metz, K.E. (1985): The Development of Children's Problem Solving in a Gears Task: A problem space perspective. *Cognitive Science 9*, 4 (October-December 1985), 431-472.

Mueller, E. (1987): *Daydreaming and Computation: A Computer Model of Everday Creativity, Learning, and Emotions in the Human Stream of Thought*, Ph.D. dissertation, University of California at Los Angeles, 1987.

Murphy, G.L., and Medin, D.L. (1985): The Role of Theories in Conceptual Coherence. *Psychological Review 92*, 3 (July 1985), 289-314.

Navinchandra, D., and Sycara, Katia, P. (1989): A Process Model of Experience-Based Design. In *The 11th Annual Conference of the Cognitive Science Society*, Cognitive Science Society, Lawrence Erlbaum, Hillsdale, NJ, August 1989, pp. 283-290.

Nielsen, P. (1989): The Role of Abstraction in Place Vocabularies. In *Proceedings of the Eleventh Annual Meeting of the Cognitive Science Society*, Ann Arbor, Michigan, August 1989, pp. 267-274.

Norman, D.A. (1983): Some Observations on Mental Models. In *Mental Models*. Lawrence Erlbaum Associates, Gentner, D., and Stevens, A., Ch. 1, pp. 7-14, Hillsdale, N.J., 1983.

Oswalt, W.H. (1976): *An Anthropological Analysis of Food-Getting Technology*, John Wiley & Sons (1976).

Pazzani, M. (1988): *Learning Causal Relationships: An Integration of Empirical and Explanation-Based Learning Methods*, Ph.D. dissertation, UCLA-AI-88-10, University of California at Los Angeles, 1988.

Quilici, A., Dyer, M. G., and M. Flowers. Recognizing and Responding to Plan-Oriented Misconceptions. *Computational Linguistics*, Vol. 14, No. 3, 1988.

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

Rosch, E. (1978): *Principles of Categorization*, Lawrence Erlbaum Associates: Hillsdale, NJ (1978).

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. *Foo* (1986), 47-73.

Smith, S.M., Glenberg, A., and Bjork, R.A. (1978): Environmental Context and Human Memory. *Memory and Cognition 6*(1978), 342-353.

Smith, S.M. (1979): Remembering In and Out of Context. *Journal of Experimental Psychology: Human Learning and Memory 5*(1979), 460-471.

Sriram, D., and Maher, M.L. (1986): The Representation and Use of Constraints in Structural Design. In *Applications of Artificial Intelligence in Engineering Problems*, Sriram, D., and Adey, R., 1st International Conference, Springer Verlag, Southampton, United Kingdom, April 1986, pp. 355-368.

Thomson, D.M., and Tulving, E. (1970): Associative Encoding and Retrieval: Weak and Strong Cues. *Journal of Experimental Psychology 86*(1970), 255-262.

Tulving, E. (1972): Episodic and Semantic Memory. In *Organization and Memory*. Academic Press, Tulving, E., , and Donaldson, W., New York, 1972.

Tulving. E. (1983): *Elements of Episodic Memory*, Oxford University Press, Oxford, UK , Oxford Psychology Series(1983).

Turner, S. (1985): MINSTREL:. In *Proceedings of the 7th Annual Meeting of the Cognitive Science Society*. Addison-Wesley, in press, 1985.

Ulrich, K. (1986): *The Use of Objects in Problem-Solving*. July 1986, Personal Communication.

Ulrich, K., and Seering, W. (1987): Conceptual Design: Synthesis of Systems of Components. In *Intelligent and Integrated Manufacturing Analysis and Synthesis*, American Society of Mechanical Engineers (ASME), ASME, December 1987, pp. 57-66.

Wasserman, K., and Lebowitz, M. (1983): Representing Complex Physical Objects. *Cognition and Brain Theory* 6, 3 (1983), 259-285.

Wilensky, R. (1983): *Planning and Understanding*, Addison Wesley (1983).

6 APPENDIX

The following articles are attached:

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.

Lange, T., Hodges, J.B., Fuenmayor, M., and Belyaev, L. (1989): DESCARTES: Development Environment for Simulating Connectionist Architectures. In *Proceedings of the Eleventh Annual Meeting of the Cognitive Science Society*, August 1989, pp. 698-705.

Device Representation for Modeling Improvisation in Mechanical Use Situations

Jack Hodges

Computer Science Department
University of California, Los Angeles

ABSTRACT

Improvisation requires an understanding and application of mechanical objects in broad contexts. The capacity to interpret a situation in terms of an object's capabilities requires the integration of functional and behavioral object representations. A model is presented which describes the integration of causal interactions between these levels of abstraction. The model maintains both intentional and behavioral representations to allow inferencing at each level, but integrates them by applying an inferencing mapping between the two. This model is used to reason about simple mechanical objects in the domain of improvisational mechanics.

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[1]. Even though a problem-solver may recognize a behavioral advantage of one object over another, their higher-level personal goals may cause them to try objects based on functional capabilities. Consider the following example of improvisation where these differences lead to a goal failure:

Broken Knife

> A man wants to polish one of his silver candlesticks. He must therefore pry open a can of silver polish in the kitchen, but doesn't want to brave winter weather to get a screwdriver from the garage. He reasons that he can use a screwdriver-like object and decides to try a carving-knife. What he doesn't realize is the knife is not strong enough in the dimension relevant for prying. The knife blade breaks.

There are many representational issues in **Broken Knife**, spanning the situational, intentional, functional and behavioral reasoning levels. At the situational level, planning choices are dictated by the relationships between the man and such contextual elements as the winter weather and objects available in the kitchen. On the intentional planning level, the man has chosen the POLISH-METALLIC plan to preserve his candlesticks. This plan requires that he have silver polish on his rag, a state which is blocked by the fact that the silver polish can is closed. Recognizing that the only resolution is to pry the can open, he realizes that the tool he usually uses for this function, a screwdriver, is in the garage. There is now a goal conflict: between his goal of preserving the candlesticks and his goal to preserve his own comfort.

Here the functional level becomes significant. A screwdriver works as a prying tool for the silver polish can because it fits into the slot between the can and lid and is strong in relation to the force necessary to pry open the lid. A carving-knife will fit into the slot and was strong enough for the functions that it was used for in the past. The knife therefore apparently matches the constraints for PRY-OBJECT, so the man uses it.

Finally there is the behavioral level. The knife is indeed strong, but only in the context of carving and along the width of the knife's blade. Along the thickness of the knife's blade, where the force of prying will be borne, the knife is not strong - not in relation to the friction force holding the lid onto the can. The knife blade therefore breaks.

We have been interested in modeling improvisation situations like **Broken Knife** in hopes of better understanding the creative process during problem-solving. Improvisation is a kind of invention where the problem-solver is constrained by circumstance. Improvisation thus encompasses the scope of EDISON, an on-going project to model the knowledge and reasoning of naive inventors, people whose knowledge and planning is based on experience rather than technical expertise [Dyer, Hodges & Flowers, 1986]. Our claim is that any approach to real-life problem-solving and decision-making must integrate each of the above levels of abstraction into a complete system. Previous object models have empha-

[1] Functional descriptions refer to the intended use of objects, as opposed to behavioral descriptions, which describe physical interactions between objects.

sized object representation at the functional or behavioral level, but none have integrated the two into a single representation and processing mechanism. EDISON has been designed to achieve this integration, and to support the associated multi-level reasoning.

REPRESENTING OBJECT FUNCTION AND BEHAVIOR

Intentional representation models have traditionally described objects with an emphasis toward their intended uses, while behavioral models have emphasized their behavioral capabilities. Each model type has been successful in its respective domain, but either could benefit from the capabilities of the other.

Intentional and Functional Object Models

Intentional object models, such as conceptual-dependency (CD) [Schank & Abelson, 1977], describe objects by an agent's intentions and how an object's function affects the outcome of those intentions (i.e. objects are black boxes). Using CD notation, the act of throwing a ball in a game of catch is represented by the thrower Propeling the ball toward the catcher while unGrasping it. The resulting state enables the ball to Ptrans from the thrower's location to the catcher's location. With this kind of model inferences can be made about the relationship between the people playing (e.g. "John threw the ball to Bill." vs. "John threw the ball at Bill."), but not about the ball involved (e.g. what if the ball never reaches Bill). This limitation presents a problem for predicting and explaining how plans are affected by object function and behavior.

Lehnert's object primitives [Lehnert, 1978] and Rieger's common sense algorithm (CSA) [Rieger, 1985] integrated object functionality into CD to describe how and when objects are used. These models introduced the idea of a functional representation level, between intentional and mechanical levels, which had properties found in both. Unfortunately, both models had weak behavioral representations and blurred the distinction between object function and behavior. They were therefore unable to take full advantage of their functional representations.

Behavioral Object Models

Behavioral object models describe objects' physical composition and interactions in lieu of their intended purpose or context. Instead of action primitives based on some form of agency, the primitives in behavioral models are simple qualitative physical process descriptions [Forbus, 1985] which describe objects and their interactions. The actor's Propel and Grasp actions (in a game of Catch) result in Force and Constraint states which enable the process, Transmit, of force to the ball. The ball unGrasping is paralleled by a Constrain process, and the resulting Force and Constraint states enable a Move process. Behavioral models are useful for predicting, explaining and simulating the ball's behavior (e.g. when the ball's weight, force and direction are known), but not for describing how or why it was thrown in the first place. Another problem with behavioral models is that they don't utilize contextual and intentional information for disambiguating, or predicting, object function during problem-solving.

Representing Objects In Edison

EDISON is an object-based representational model for reasoning about situations like **Broken Knife** by integrating object knowledge derived from intentional and behavioral points of view. The intentional part of EDISON'S bi-level model considers the object as an instrument to achieving specific goals in specific contexts. The behavioral part of EDISON considers the object and its behavioral dependencies with other objects. This integration is achieved by considering the structural continuity which must be maintained to support inferences between these abstraction levels, and by considering a third, *functional*, part which overlaps the intentional and behavioral abstraction levels and provides for a continuous inference path between them.

The objects described in EDISON are simple mechanical devices, such as screwdrivers, hammers, knives, can openers, and nail clippers. In EDISON, the representational emphasis is on the physical qualities and relations which support an object's functional description. Most of the reasoning in EDISON is done at higher levels, so the simulator is only used for diagnosis and explanation. This contrasts to detailed qualitative simulators, such as [Doyle, 1988], designed for this purpose. The EDISON model represents all objects as combinations of primitive devices (such as levers, springs, and wheels) which effect the leverage mechanism through the Transform process [Hodges, Dyer & Flowers, 1987. All object behavior can thus be described in terms of the transmission, translation, or magnification of force and motion.

Object *functions* refer to the tasks an object has been or could be applied to in a particular context, and have both intentional and behavioral qualities. Using a knife to carve turkey, to threaten someone, to tighten screws, or to pry can lids all describe knife functions. At the intentional level object functions describe this context sensitivity through *attributes*, which are qualities associated with an object's functional capability relative to other objects. For example, if we want to carve a turkey, then we need an object which has a *sharp* and *long* blade relative to the turkey.

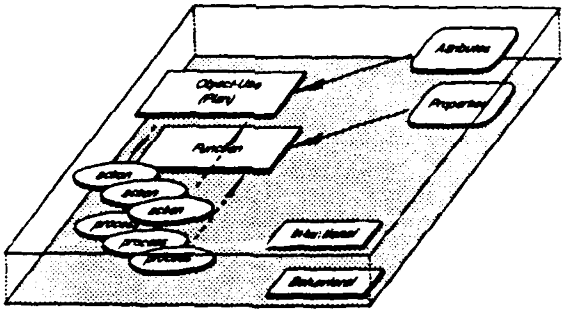


Figure 1: Knowledge structures and their causal relationships are isomorphic for intentional and mechanical representations.

An object's attributes direct planning choices in context by constraining applicable functions. At the behavioral level functions organize the processes (as process-state sequences) which effect the object's behavior. Processes are constrained by an object's physical properties and its relationships with other objects.

Intentional-Behavioral Representational Continuity

The relationship that object function plays in integrating intentional and behavioral models is depicted in figure 1. Whether viewed intentionally or behaviorally, the same object function is represented in a given situation. Each point of view provides different inferences about the object, so in EDISON the causal relationships are kept distinct. For example, in the game of catch we may want to make inferences about the ball Ptransing (such as why it was thrown), or its Moving (such as how and where it will go), depending on our goals. If we simply merge the representation levels one set of inferences is lost.

It is also important to remember that plans and functions in a given situation both describe the same behavior, but simply at different levels of abstraction. In EDISON these relationships are maintained by a structural isomorphism between intentional and behavioral knowledge structures. For example, consider the *plan-action-state* relationship which describes causality at the intentional level. This has a one-to-one correspondence with the *function-process-state* relationship at the behavioral level.

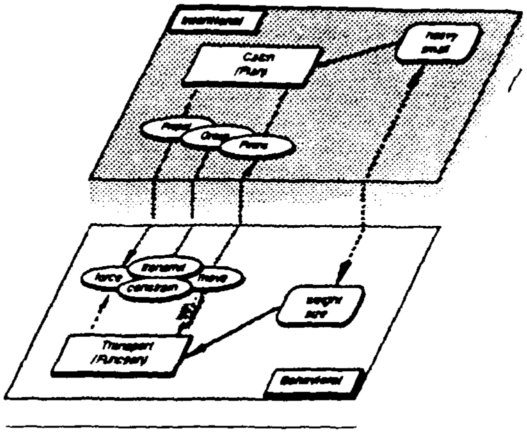


Figure 2: The bi-level representation for a game of catch shows the continuous, albeit separate, inference path between intentional and mechanical abstraction levels.

The bi-level model is designed to describe situations like the game of catch introduced above and depicted in figure 2. The intentional representation is shown on the upper level and the behavioral representation is shown on the lower level. The intentional description has a causal "gap" after the thrower's unGrasp action, whereupon the ball Ptranses to the catcher. The behavioral representation overlaps at this point, with the enabling and constraining conditions for the Transport function, and continues to describe the ball's behavioral path (paralleling the Ptrans action) until the Transport function terminates (i.e. the ball's motion ceases). The Transport terminating state is identical to the Catch plan's resulting state (arrival at the intended location). By integrating intentional and behavioral representations this way inferences can be made about object function and behavior not possible with either level alone.

Intentional-Behavioral Inference Continuity

There is a difference in generality between intentional and behavioral reasoning levels which, despite the structural continuity, obviates direct inferences between the two levels. However, because the same object is considered at both inference levels, its functions provide the necessary inference continuity through the associated constraining attributes and properties.

At the intentional level, attributes describe functional capabilities of an object learned through experience, and are specific to particular objects in particular contexts. Knowing the attribute enables high-level inferences about its functional capabilities if the context is reinstated. For example, knowing that a carving-

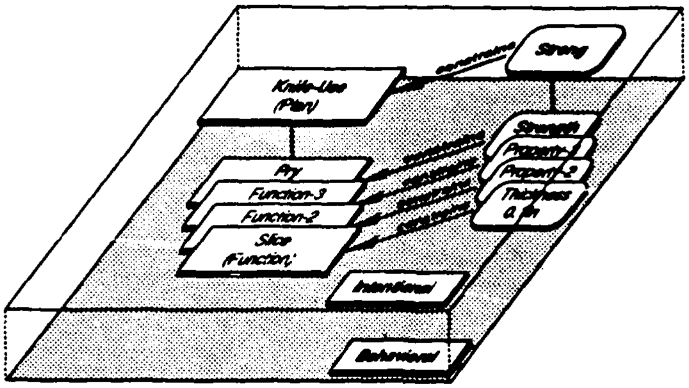


Figure 3: Functional attributes like *strong* are causally related to plan application through the constraints they place on object functionality. Likewise property values constrain the underlying processes. Different attributes will be associated with different situations, and different property values will support the associated functions.

knife was successful in transmitting force for carving a turkey, one might have concluded that the knife is a strong object (w.r.t. the turkey). The *strong* attribute of the knife is a relative term between like property values of the knife and bird, and is only valid for this situation. Other situations requiring *strong* objects, however, might remind the problem-solver of the carving-knife. Attributes thus affect planning, providing grist between context and a problem-solver's associated interpretation. Figure 3 depicts the relationship between different attributes, such as *strong* and *thin*, and how they constrain Knife-Use via the knife functions Pry-Object and Slice.

At the behavioral level the *strong* attribute is associated with the knife's value for the breaking-strength[1] property, which directly constrains the Pry-Object function's processes. Knowing the knife's value for breaking-strength guarantees inferences about its capacity to pry. The correspondence between the *strong* attribute and the breaking-strength property value enables inferences between levels. The difference between object functionality based on the attribute, *strong*, and that based on the property, breaking-strength, is that dimensionality (i.e. detail) is lost. If the problem-solver retrieves the knife based on the higher-level functionality (for example during planning), then the dimension of strength is unlikely to be remembered. In Broken Knife this leads to failure. However, the fact that a screwdriver was *strong* for its intended function for tightening screws, leads to an inference that it will be *strong* for other functions as well, such as prying a varnish-can or punching an oilcan for which it is an effective tool. If the knife's behavior is the basis for retrieval (for example during problem-solving experimentation), then dimension is remembered and predictions, inferences, or explanations can be made with confidence.

Attribute-Property Relationships

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. If the correspondence is made, then the inferences can be compared and behavior modified. Each attribute defines a *range* in a property's quantity space. The two attributes, *light* and *heavy*, which describe the weight property of an object, illustrate a many-to-one relationship which is characteristic between attributes and property values. Many attributes are associated with object function through a specific property, such as *strong* to strength, or *long* to length. Attributes can also be described by combinations of properties or other attributes. The attribute *metallic*, for example, is described by the attributes *shiny*, *smooth*, *cold* and *hard*.

There are no exact correspondences between an attribute and its associated property value, since attributes are context-dependent. Nevertheless, some comparisons can be made based on how properties and attributes are represented. In EDISON property values are defined as (property, dimension, value) triples, and attributes as (property, reference) doubles. These relationships are illustrated in figure 4 for the carving-knife's *strong* attribute in **Broken Knife**.

[1] The equivalent force an object can withstand prior to failure.

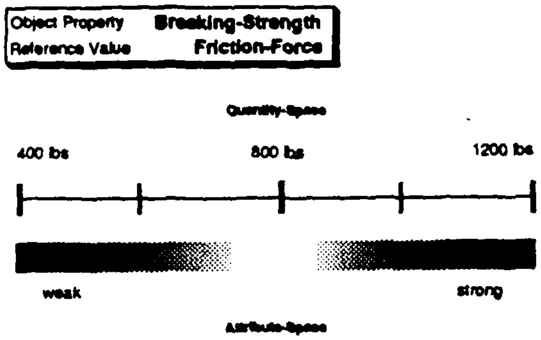

Object Property Breaking-Strength
Reference Value Friction-Force
Quantity-Space
400 lbs
800 lbs
1200 lbs
weak
strong
Attribute-Space

Figure 4: The attributes *weak* and *strong* illustrate the relative breaking-strengths (shown in pounds) of objects and their context dependency.

The attributes *weak* and *strong* map onto the material proper'v describing breaking-strength. The numbered line segment in figure 4 represents a portion of quantity space describing breaking-strength values, with its central value being the Friction-Force attribute reference. There are two ways that attributes are referenced to property values in quantity-space:

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

The reference point defines the context which an attribute is directly applicable, and is always found in the situational context. For instance, in **Broken Knife** the reference can be the can's or screwdriver's, breaking-strength, or the friction force between the Canlip and the Lidlip. Either way the knife is comparably weak. The shaded bars in figure 4 represent attributes, and show the variation of the terms *weak* and *strong* with respect to Friction-Force. The shading indicates the generalized relationship between what the man in **Broken Knife** knows about knife and can strength.

The attribute-property value relationship combined with the bi-level structural isomorphism provides a continuous inference path between intentional and behavioral levels of abstraction. If a situation exists in memory where a carving knife has successfully been used as a *strong* object, say to cut meat, then EDISON will likely try to use it again when the need for a *strong* object arises (e.g. in **Broken Knife**).

REASONING ABOUT OBJECTS IN CONTEXT

The **Open:varnish-can** situation shown in figure 5 illustrates the effect that attributes have at the intentional level. The associated property value and behavioral effects have also been depicted in figure 5, but a detailed description can be found in [Hodges, 1989]. At the intentional level varnishing a chair entails a number of preparatory steps, one of which is to get the varnish onto a paintbrush (a D-Cont goal). In figure 5 this step leads to a plan for opening the varnish can by prying the lid with the tip of a screwdriver.

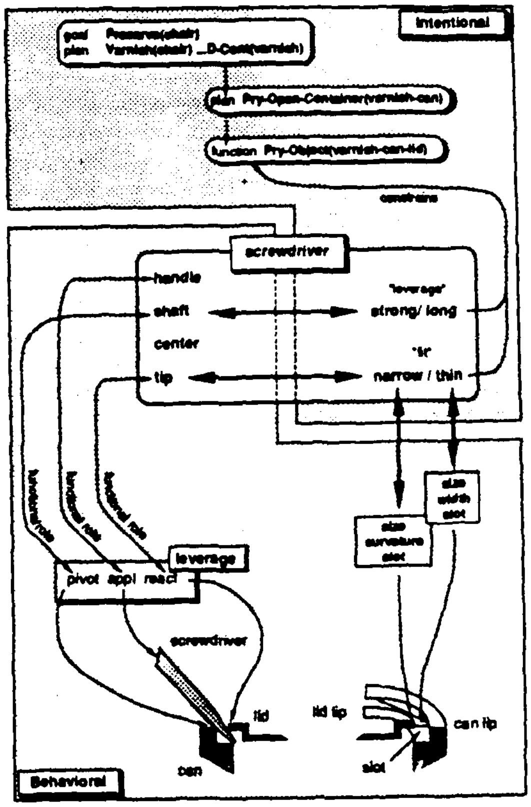


Figure 5: The attributes associated with "leverage" and *"fitting" are instrumental in representing* how a screwdriver is used to pry a varnish-can lid in **Open:varnish-can** by constraining its application. Attributes map both to object regions (such as *handle, shaft* and *tip*) and property values, which constrain functional interactions.

At the functional level the Pry-Object function is governed by two attribute groups, one for leverage and one for fit. The "leverage" requirement states that the screwdriver be *long*, so that sufficient mechanical advantage can be gained to overcome the friction holding the can and lid together, and *strong* so that it won't break under this force. The "fit" requirement states that the tip of the screwdriver must be *thin* and *narrow* compared to the slot between the canlip and lidlip, and constrains the Contact and Magnify processes in Pry-Object at the behavioral level. The leverage box in Figure 5 states that any object with regions of force application, pivot, and force reaction can be used to apply leverage. The screwdriver has these regions bound to its handle, shaft, and tip. In terms of prying these screwdriver regions are the only locations of interest. There are similar regions associated with the can (i.e. the lid, lidlip, can, and canlip). The regions on both objects are also used to define the attribute reference points for prying.

Bi-Level Representation and Situation Interpretation

The primary reason for describing object use at varying abstraction levels is to support different object interpretations depending on context. We want a representation model which describes how a screwdriver or knife is used as a utensil in one circumstance, a weapon in another, and a paperweight in a third. Each of these situations calls upon the same object property (weight), but with different required property values. Models that are context independent bar behavioral descriptions from addressing an actor's perspective in the same way that models that are context sensitive bar a functional description from making predictions about behavior. However, even when an object has only been used in a single context (such as using a carving knife for slicing), the attributes which enabled its functionality might enable its use in other contexts.

Knife breaking-strength provides a good example of this. Objects used to cut must be *strong* enough that they do not bend or break before the cut is completed. Of course, knife strength is only meaningful in the dimension of the intended cut. However, a person who naively uses a knife might generalize the extent of strength to all of its dimensions.

Figure 6 illustrates how **Broken Knife** is represented at the situational level. The upper window illustrates the information given. The lower window illustrates a number of situations where simple devices have been used in standard ways, and the attributes which [illegible]ain their fu~~ctionality. The D-Cont goal to [illegible] polish onto a rag leads to an Open-Cont[illegible]. This information is provided to memo[illegible] a retrieval cue. The **Open:varn[illegible]** situation, where a screwdriver is used for prying, is the best functional match but conflicts with the man's P-Comfort goal. The result is that screwdriver-use, and screwdriver-related experiences, are unavailable for planning (with a screwdriver). This is shown with circle-ended dotted lines. The screwdriver attributes which are pertinent to prying are shared (situationally) with other devices (e.g. carving knife in **Slice:turkey**) which can be applied to the Pry-Object function. The **Flip:pancakes** situation is inappropriate because the spatula has attribute *broad*, which conflicts with the *narrow* attribute instrumental to Pry-Object. The carving-knife is also applicable based on availability, since the carving-knife resides in the kitchen setting of **Broken Knife**. The end result is a plan combining the Pry-Object function with the carving-knife object.

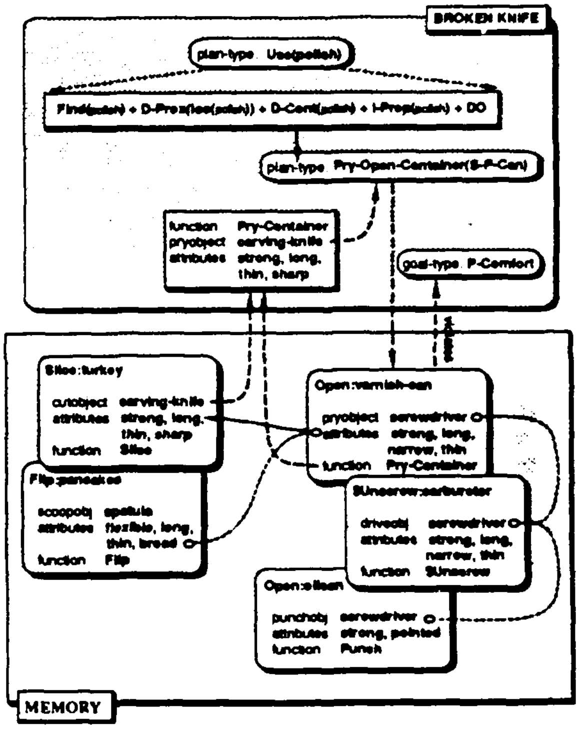


Figure 6: The **Broken Knife** situation illustrates situational interpretation of a carving knife based on its *strong* attribute. The Open-Container(S-P-Can) plan is indexed in memory to situations where objects have been used for opening. **Open:varnish-can** is strongly associated but cannot be applied directly because of a goal conflict. The screwdriver and carving knife share attributes instrumental to prying, so that an alternate Pry-Object plan using the carving knife can be applied to the situation.

A DETAILED EXAMPLE

The representation for the functional and behavioral inference paths in **Broken Knife** in figure 6 is fleshed out in figure 7. The behavioral description shown in figure 7 represents the process interactions supporting the Pry-Object function with the knife instantiated as the Pry-Object. The representation is shown instantiated with the carving knife after retrieval from memory and combination into the Pry-Object function. The attribute/property-value rela-

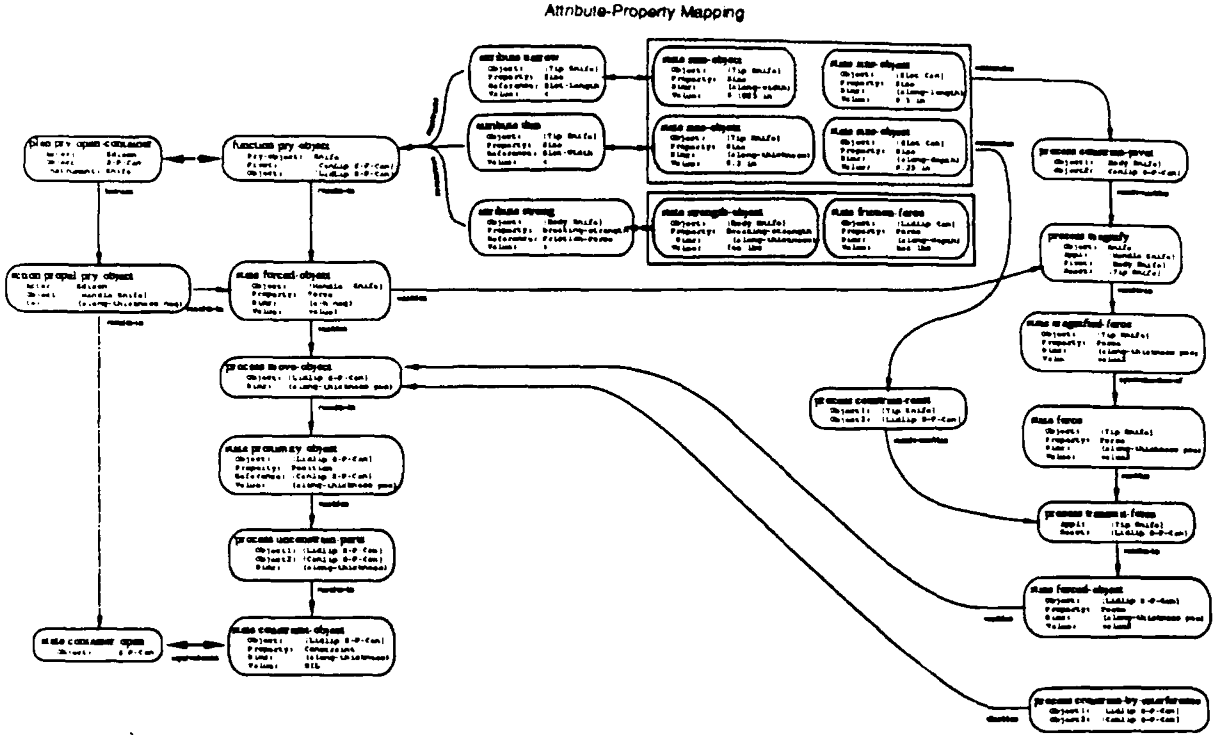


Figure 7: Functional-behavioral representation for **Broken Knife**. The attribute-property relationships constrain the Pry-Object function and the processes which comprise it. Attributes are associated with an object in context so the carving knife *strong*, *narrow* and *thin* attributes are associated with a retrieved situation, **Slice:turkey**. Some of the fillers illustrated (e.g. [Lidlip S-P-Can]) are simplifications of the actual representation.

tionship is shown as it affects the functional/behavioral description under the heading Attribute-Property Mapping. The *fit* requirement affects Pry-Object in two dimensions, so the comparison to size is made in two dimensions. The size values constrain the processes Magnify and Transmit-Force. The darkened two-way arrows between attributes and property values (states) represent a "many-to-one" link. The dashed and darkened two-way arrow between Pry-Object and Open-Container illustrates the inference cross-over between the functional and behavioral level.

The planning and interpretation involved in **Broken Knife** and the other situations illustrated in figure 6 are currently being implemented in ROBIN, a localist spreading-activation model of high-level inferencing [Lange & Dyer, 1989], which uses the DESCARTES connectionist simulator [Lange, Hodges, Fuenmayor, & Belyaev 1989]. In this implementation there will be equivalent inference paths for other devices which could be used as the Pry-Object filler, such as the candlestick itself. These inferences compete with the use of the knife through the spread of activation. The carving knife inference path will win out and be chosen as the plan for prying open the container, however, since its *strength* and *fit* attributes match the constraints on the Pry-Object role better than the other available objects.

CONCLUSIONS

Designing a knowledge representation model which supports the invention process requires an integration between intentional and behavioral object descriptions. The model must address how the environment and people's higher-level goals and intentions affect object choice during problem-solving, and how objects' properties support that functionality at the behavioral level. The bi-level representation used in the EDISON model provides the necessary integration and maintains the inferences from each abstraction level. The concept of attributes is introduced, and their relation to property values is discussed with respect to how they affect inferences between intentional and behavioral levels of abstraction.

Acknowledgements

This research has been supported by a contract with the Office of Naval Research, contract number N00014-86-0615. The author expresses thanks to Trent Lange for invaluable discussion and assistance in the preparation of this documenL Thanks also to Michael Dyer, Colin Allen, and the anonymous reviewers for their editorial assistance.

References

DeKleer, J. & Seely-Brown, J.S. (1985): Qualitative Reasoning About Physical Systems, edited by Daniel G. Bobrow, MIT Press, pages 7-84.

Doyle, R.J. (1988): *Hypothesizing Device Mechanisms: Opening Up the Black Box*, MIT Artificial Intelligence Laboratory TR 1047.

Dyer, M., Hodges, J.B., & Flowers, M. (1986): EDISON: Engineering Design Invention System Operating Naively, *Journal of Artificial Intelligence in Engineering, Vol. 1 No. 1*, p. 36-44.

Forbus, K. (1985): Qualitative Reasoning About Physical Systems, edited by Daniel G. Bobrow, MIT Press.

Hodges, J.B. (1989): *Foundations for Creativity: Integrating Functional and Behavioral Object Representations for Problem-Solving*, Ph.D. Dissertation, Computer Science Department, University of California at Los Angeles (forthcoming).

Hodges, J.B., Dyer, M. G., & Flowers, M. (in press): Knowledge Representation for Design Creativity. In D. Sriram and C. Tong, editors, *Artificial Intelligence Approaches to Engineering Design*, Addison-Wesley, (in press).

Lange, T. & Dyer, M. G. (1989): Frame Selection in a Connectionist Model of High-Level Inferencing. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society (CogSci-89)*, Ann Arbor, MI, August 1989.

Lange, T., Hodges J., Fuenmayor, M., & Belyaev, L. (1989): DESCARTES: Development Environment For Simulating Hybrid Connectionist Architectures. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society (CogSci-89)*, Ann Arbor, MI, August 1989.

Lehnert, Wendy (1978): The Process of Question Answering. Lawrence Erlbaum Associates. Chapter 10.

Rieger, Chuck (1975): In *An Organization of Knowledge for Problem Solving and Language Comprehension*, Morgan Kaufmann, p. 487-508.

Schank, R. C. & Abelson, R. (1977): *Scripts, plans, goals and understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Program of the

Eleventh Annual Conference of the Cognitive Science Society

16-19 August 1989
Ann Arbor, Michigan

LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
1989 Hillsdale, New Jersey Hove and London

DESCARTES: Development Environment for Simulating Hybrid Connectionist Architectures

Trent E. Lange, Jack. B. Hodges, Maria. E. Fuenmayor, Leonid. V. Belyaev

Computer Science Department
University of California, Los Angeles

ABSTRACT

The symbolic and subsymbolic paradigms each offer advantages and disadvantages in constructing models for understanding the processes of cognition. A number of research programs at UCLA utilize connectionist modeling strategies, ranging from distributed and localist spreading-activation networks to semantic networks with symbolic marker passing. As a way of combining and optimizing the advantages offered by different paradigms, we have started to explore hybrid networks, i.e. multiple processing mechanisms operating on a single network, or multiple networks operating in parallel under different paradigms. Unfortunately, existing tools do not allow the simulation of these types of hybrid connectionist architectures. To address this problem, we have developed a tool which enables us to create and operate these types of networks in a flexible and general way. We present and describe the architecture and use of DESCARTES, a simulation environment developed to accomplish this type of integration.

INTRODUCTION AND MOTIVATION

Within the connectionist approach there are three paradigms, each having its own advantages and disadvantages: Distributed Connectionist Networks (DCNs), Localist Connectionist Networks (LCNs), and Marker-Passing Networks (MPNs).

DCNs (such as the models in [Rumelhart & McClelland, 1986]) use simple, neuron-like processing elements which represent knowledge as distributed patterns of activation. DCNs, sometimes known as *Parallel Distributed Processing* or *subsymbolic* models, are interesting because they have learning rules that allow stochastic category generalization, they perform noise-resistant associative retrieval, and they exhibit robustness to damage. Distributed models, however, have (so far) been sequential at the knowledge level, lacking both the structure needed to handle complex conceptual relationships and the ability to handle dynamic variable bindings and to compute rules.

LCNs (as exemplified by the models of [Waltz & Pollack, 1985] and [Shastri, 1988]) also use simple, neuron-like processing elements with numeric activation and output functions, but represent knowledge using semantic networks of conceptual nodes and their interconnections. Unlike DCNs, localist networks are parallel at the knowledge level and have structural relationships between concepts built into the connectivity of the network. Unfortunately, they lack the powerful learning and generalization capabilities of DCNs. They also have had difficulty with dynamic variable bindings and most other capabilities of symbolic models.

MPNs (as exemplified by the models of [Charniak, 1986] and [Hendler, 1988]) also represent knowledge in semantic networks and retain parallelism at the knowledge level. Instead of spreading numeric activation values, MPNs propagate symbolic markers, and so support the variable binding necessary for rule application, while preserving the full power of symbolic systems. On the other hand, they do not possess the learning capabilities of DCNs or exhibit the inherent evidential constraint-satisfaction capabilities of LCNs.

Hybrid Connectionist Models

Research at UCLA has spanned the range from subsymbolic to symbolic connectionist models [Dyer, 1989]. A number of us have begun to construct hybrid architectures which use what we term Multiple Interacting Networks, or MINs, heterogeneous connectionist networks that communicate via shared elements. A neurophysiological approach [Nenov & Dyer, 1988] effectively uses MINs for visual/verbal association by modeling heterogeneous neuronal characteristics in separate networks. We have also been exploring the use of MINs for higher cognitive tasks, such as planning, creativity, story invention, and political negotiations. In political negotiations research, for instance, MINs are used to simulate the multiple perspectives of negotiating parties.

Another approach is to build models that combine the bottom-up processing features of DCNs with the top-down processing features of LCNs and MPNs. Figure 1 shows **Hiding Pot**, an example wherein elements from each paradigm are combined using MINs. This allows us to approach a problem that would be difficult, if not impossible, using a single paradigm. **Hiding Pot** shows a simplified network built to understand the sentence, *"John put the pot inside the dishwasher because the police were coming."*[1] Network-A in Figure 1 utilizes an MPN to do role-binding and an LCN to activate and combine evidence for individual schemas. These then combine their functionality to support predictions and perform inferencing and disambiguation.

One might also want to combine different connectionist approaches by having separate networks that communicate with each other, where each one performs a different cognitive task. Network-B in Figure 1 is a DCN, trained to recognize words from line segments [McClelland & Rumelhart, 1986, chap. 1]. By integrating these two approaches, we can simulate cognitive processes at the different levels of abstraction necessary for modeling reading and understanding.

Network-A interacts with Network-B through shared lexical nodes. Once a word has been recognized, it passes activation to the concepts related to the word. For example, the node for concept John gets activation from the word node "john" which is shared by both networks. Activation then propagates along the chain of related concepts in the network as contextual evidence for disambiguation. Markers are passed over the role nodes across marker passing links between corresponding roles to represent role-bindings and perform the needed inferencing.

While there are several existing connectionist simulators, none allows the simulation of multiple interacting hybrid networks, as in **Hiding Pot**, that integrate elements from more than one paradigm of connectionist modelling. We have developed the DESCARTES *simulation environment* specifically to address this kind of integration. DESCARTES enables researchers to design, simulate, and debug hybrid connectionist architectures that combine elements of distributed, localist, and marker-passing networks.

DESCARTES ARCHITECTURE

DESCARTES is a package designed for simulating network processing, network interaction, and integration of networks into an overall processing environment. The system consists of two interactive components: network *elements*, such as nodes and links, their associations, and their functionality, and *processing controllers*, which organize network elements and coordinate their processing. The components of this architecture, as applied to **Hiding Pot**, are shown in Figure 2.

[1] The inferencing and frame selection needed to understand sentences such as **Hiding Pot** is explained more thoroughly in [Lange & Dyer, 1989a] and [Lange & Dyer, 1989b], which describe ROBIN, a model of high-level inferencing using an LCN without marker-passing.

Hiding Pot

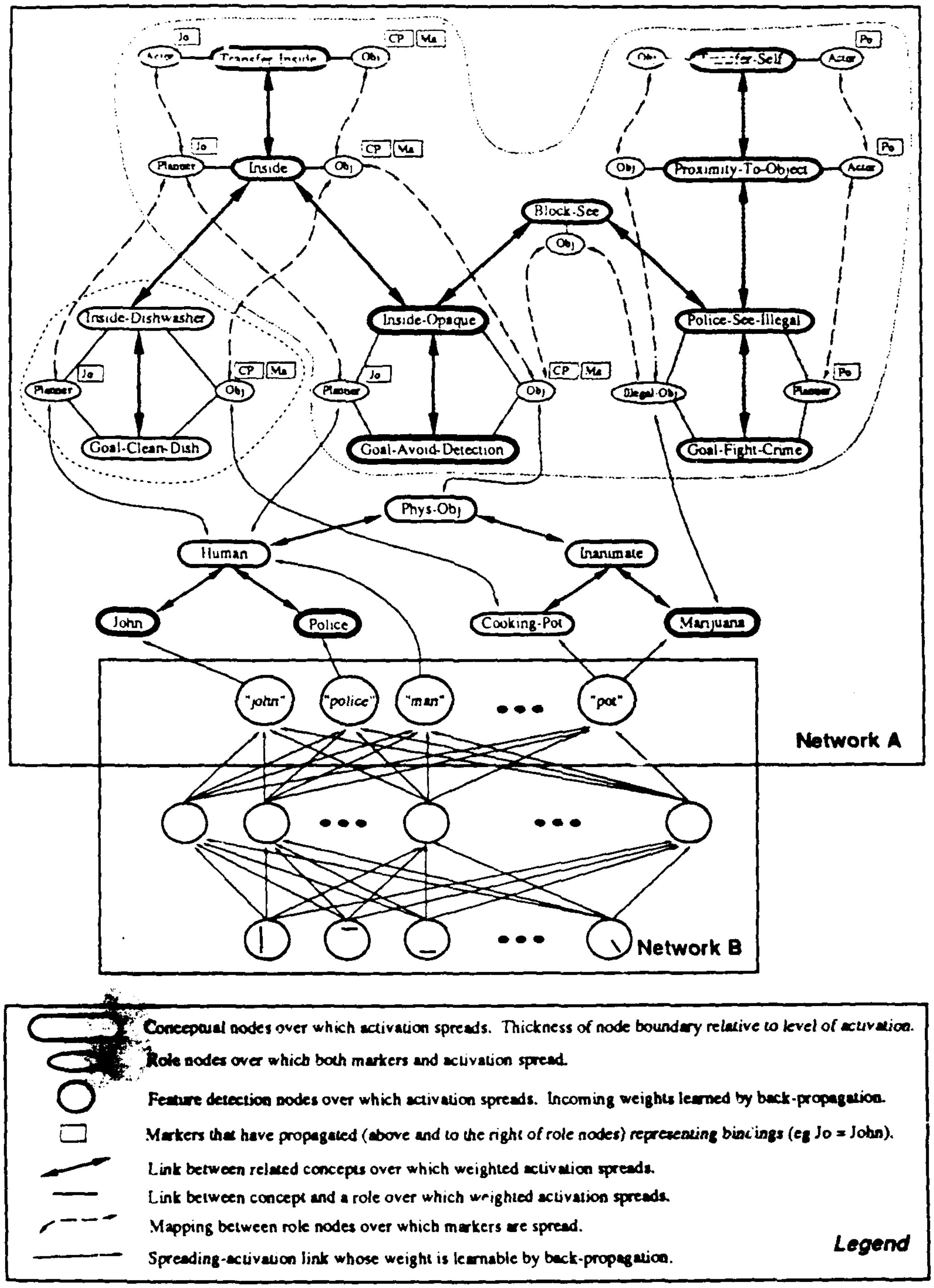

Jo
CP
Ma
Actor
Transfer-Inside
Obj
Planner
Inside
Obj
Transfer-Self
Pe
Actor
Obj
Proximity-To-Object
Actor
Block-See
Obj
Inside-Dishwasher
Inside-Opaque
Police-See-Illegal
Planner
Obj
Planner
Obj
Illegal-Obj
Planner
Goal-Clean-Dish
Goal-Avoid-Detection
Goal-Fight-Crime
Phys-Obj
Human
Inanimate
John
Police
Cooking-Pot
Marijuana
"john"
"police"
"man"
"pot"
Network A
Network B
Conceptual nodes over which activation spreads. Thickness of node boundary relative to level of activation.
Role nodes over which both markers and activation spread.
Feature detection nodes over which activation spreads. Incoming weights learned by back-propagation.
Markers that have propagated (above and to the right of role nodes) representing bindings (eg Jo = John).
Link between related concepts over which weighted activation spreads.
Link between concept and a role over which weighted activation spreads.
Mapping between role nodes over which markers are spread.
Spreading-activation link whose weight is learnable by back-propagation.
Legend

Figure 1: The sentence *"John put the pot in the dishwasher because the police were coming."* illustrates the utility of integrating semantic networks (Network-A) and distributed networks (Network-B). The darkest area represents the most highly-activated set of nodes representing the network's plan/goal analysis of the sentence. Not all markers are shown. Location role nodes and other parts of the network are also not displayed.

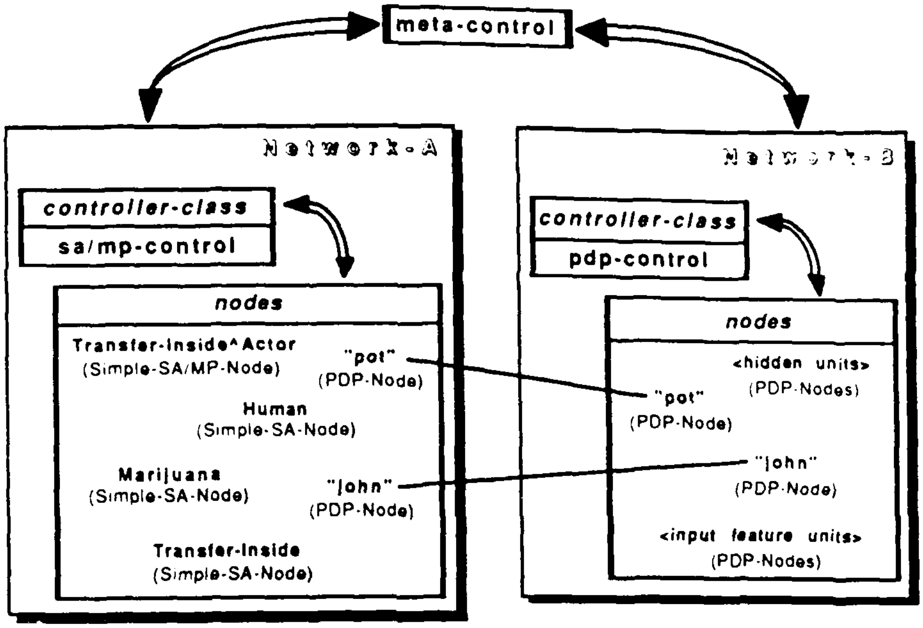


Figure 2: DESCARTES Processing Architecture applied to **Hiding Pot**. Shown in each network are a few of their nodes, with the class of each node being declared in parentheses below their names. PDP-Nodes "pot" and "john" are shared by both networks.

Processing Controllers

When DESCARTES is loaded and running, the required processing controllers are a *meta-controller* (a supervisor for all elements and sub-controllers present in the run-time system) and at least one *network controller* (a supervisor for an individual network and its elements). The architecture described in Figure 2, and implemented in **Hiding Pot**, is controlled by a meta-controller (Meta-Control) which coordinates the two networks (Network-A and Network-B). Each of these networks has a local network controller which coordinates the processing of its elements. In this case the controller for Network-A is of class SA/MP-Control, which combines both spreading-activation and marker-passing functionality.

Network Elements

The nodes shown in **Hiding Pot** are illustrative of the kinds of nodes provided in the system. Three of DESCARTES's predefined node classes are used in **Hiding Pot**: (1) Simple-SA-Node, used in **Hiding Pot** for conceptual elements, such as Human and Transfer-Inside, (2) Simple-SA/MP-Node, used for roles, such as Transfer-Inside^Actor, and (3) PDP-Node, used for feature detection in Network-B, such as the node representing lexical entry "pot". Figure 3 provides an example of node creation in DESCARTES.

Simple-SA-Node is a basic class of spreading-activation nodes with default activation and output functions. Simple-SA/MP-Node is another standard node class, which combines the functionality of Simple-SA-Node with that required for marker passing. Finally, PDP-Node is the simplest class of DCN-type nodes — spreading-activation nodes that modify the weights on their input links by backpropagation [Rumelhart *et al.*, 1986, chap. 8].

Many other common node and link types are predefined, with a variety of activation, threshold, and output functions. More complicated classes are also available, including gated nodes and

```
(Simple-SA-Node Transfer-Inside :in-links (SA-Link ("put"                  0.75)
                                                   (Inside                 1.00)
                                                   (Transfer-Inside^Actor  0.50)
                                                   (Transfer-Inside^Obj    0.50)
                                                   (Transfer-Inside^Loc    0.50)))

(Simple-SA/MP-Node Transfer-Inside^Actor :in-links (SA-Link (Transfer-Inside 1.0))
                                                   (MP-Link Inside^Planner))
```

Figure 3: Creation of Transfer-Inside and Transfer-Inside^Actor nodes, with forward-referencing.

links, along with more neurally-realistic nodes that communicate via output spikes, such as the artificial neural oscillators of [Vidal & Haggerty, 1987]. The functionality of DESCARTES objects can easily be extended by combining the default class definitions of the object hierarchy with user-defined modifications, a process described in [Lange *et al.*, 1989].

Structured Networks

Some connectionist models have a consistent structure between groups of nodes in the network. In a semantic network, for example, a node representing the head of a frame might always be connected via a certain type of link to each of its roles, which in turn might always have a node for their fillers. Groups of nodes forming winner-take-all networks are always completely interconnected with constant inhibitory weights. Rather than force the user to repetitively define all nodes and connections for each such structured group, DESCARTES has a facility that allows the programmer to optionally define a structured growing method for each node class. A node's growth method automatically creates the node's expected structured incoming and outgoing nodes and connections. This feature allows knowledge base definitions to act as keys for network creation rather than as exhaustive listings of the networks' nodes and their connectivity.

SIMULATION IN DESCARTES

Once the networks have been designed and built, the user starts the simulation by (1) optionally defining the cycling, termination, and display sequence for each network, (2) initializing the meta-controller to clear out all activation and markers, (3) activating or marking the desired nodes, and (4) starting the cycling sequence and specifying the number of global cycles to run. An example of this process is shown in Figure 4, but for a complete description see [Lange *et al.*, 1989].

Figure 4 shows the initial activation and markers needed to process the phrase *"John put the pot inside the dishwasher."* The first define-cycling command in the figure specifies that the meta-controller spread activation in Network-A once per global cycle, while only passing markers once per every three global cycles. Both activation and markers will cycle until stability, their default termination condition. For analysis of the network's activity, the user has defined that a trace of the markers' propagation be shown and that the status of the nodes be displayed every ten cycles. The second define-cycling command defines that Network-B is not to be cycled in this example.

In general, the networks' cycling sequences need only be set once per session (if at all), although all sequencing and displaying parameters may be re-specified in mid-simulation. Activations and markers of nodes may be changed at any time. The cycling sequence is further described below.

The Simulation Cycle

As shown, DESCARTES is designed in such a way that networks can be cycled in parallel or serially. The meta-controller provides for timing coordination between the networks. Networks cycled in parallel behave as if they were a single net, even though they need not operate at the same fre-

```
(define-cycling %Network-A :sa-cycle-every     1    ;; (1)
                           :marker-cycle-every 3
                           :marker-trace       T
                           :display-every      10)
(define-cycling %Network-B :sa-cycle-every     NIL)
(init meta-control)                                 ;; (2)

(clamp-activation %"put"      1.0)                  ;; (3)
(mark %Transfer-Inside^Actor (marker %John))
(mark %Transfer-Inside^Obj   (marker %Cooking-Pot)
                             (marker %Marijuana))
(mark %Transfer-Inside^Loc   (marker %Dishwasher))

(cycle 50)                                          ;; (4)
```

Figure 4: An example of the DESCARTES control language.

quency, or, in fact, with the same functionality. A particular model may have a network of inhibitory nodes cycling at a faster rate than a network of excitatory nodes with which it interacts, at the same time as symbolic markers are being passed over each, and backpropagation is being performed within sub-networks of the model. With serial cycling, one network may wait until another network completes a specified number of cycles or reaches stability before starting to cycle itself.

Each global network cycle is comprised of four steps: (1) determination of which networks need to be cycled, (2) update of active nodes in the cycling networks, (3) spread from active nodes in the cycling networks to their out-links, and (4) report any requested output.

Determining Active Networks: The meta-controller determines which of the networks in the system need to be cycled in parallel on the given cycle, according to defaults and any define-cycling commands. In Figure 4, spreading-activation nodes in Network-A will be cycled on every global cycle, while marker-passing nodes will be cycled only on global cycles 1, 4, 7, and so on, until termination (stability).

Update: Each active node in the cycling networks queries its incoming links for new activation and/or markers. Spreading-activation nodes calculate their new activation by applying their activation function, while marker-passing nodes store any new markers they have received.

Spread-To-Out-Links: Each active node in the cycling networks calculates its output (either activation or markers) and sends it to its outgoing links. The output of spreading-activation nodes is calculated by applying their output function, while the output of marker-passing nodes is generally their new markers.

Report Output: The final step of a cycle entails querying the cycling networks for results. Each network controller can optionally display the status of important nodes at specified cycles (Network-A's status will be displayed every 10 cycles in Figure 4) or trace new activation and/or markers. DESCARTES currently has a number of output options useful for system design and debugging.

IMPLEMENTATION AND SIMULATOR ACCESS

DESCARTES has been designed for portability, flexibility, and simplicity of use. Portability is achieved via the use of COMMONLISP, the ANSI Lisp standard. Flexibility is augmented by the use of the COMMONLISP Object System, CLOS, whose hierarchical class structure provides inheritance which enables users to utilize pre-defined functional classes to customize their own semantics. A complete description of currently available functionality and test-bed cases can be found in [Lange *et al.*, 1989]. The largest test case simulated to date is an implementation of a ROBIN [Lange & Dyer, 1989b] network in the domain of **Hiding Pot**. It consists of two interacting LCNs built from four node classes and five link classes, with a total of 12,400 nodes and 40,000 links.

DESCARTES's control language is simple and effective, enabling the designer to easily set up and test different network configurations using either pre-defined or user-defined elements. At the same time, the system has been designed with ease of network debugging in mind, with history and output facilities that offer researchers valuable methods for interpreting network behavior.

DESCARTES will be made available to all interested users. Enquiries about access to the simulator should be sent to DESCARTES@CS.UCLA.EDU.

RELATED WORK

Some of the recent tools constructed for building and simulating connectionist architectures are (1) the Rochester Connectionist Simulator (RCS) [Goddard *et al.*, 1987], (2) the PDP Software Package [McClelland & Rumelhart, 1988], (3) MIRRORS/II [D'Autrechy *et al.*, 1988], and (4) GENESIS [Wilson *et al.*, 1988]. RCS is a spreading-activation simulator which allows units to have any amount of associated data. There is no specification language for construction of the net, but the system provides a library of commonly used network structures and units. The PDP Software package includes a number of programs for simulating the DCN models in [Rumelhart & McClelland, 1986]. MIRRORS/II and GENESIS, the most recent of the four systems, have both features: a high level non-procedural language for network construction and an indexed library of commonly used networks. Both have more sophisticated and flexible control mechanisms than RCS and the PDP Software Package, with MIRRORS/II emphasizing simulations using LCNs and GENESIS emphasizing realistic, biologically-based models.

The flexibility and symbolic capabilities afforded by DESCARTES' object-oriented implementation in COMMONLISP and CLOS comes at a small expense in simulation speed in comparison to the C-based implementations of RCS, the PDP package, and GENESIS. The only case where the difference in speed should be significant, however, is in simple backpropagation networks requiring thousands of learning epochs, for which the PDP package might be more appropriate. Except for GENESIS, all of the above-mentioned simulators are geared toward monotonic distributed or localist spreading-activation networks. None of them have the concept of hybrid multiple interactive networks as part of their design, especially those which can pass symbolic markers.

CONCLUSIONS

We have presented a development tool, DESCARTES, which provides researchers with the capability to combine Distributed Connectionist Networks, Localist Connectionist Networks and Marker-Passing Networks within a single simulation environment. The most important theoretical contribution of DESCARTES is the concept of Multiple Interactive Networks with intra- and inter-network heterogeneity. As a tool, it provides a simple, portable, and versatile environment for designing and testing different cognitive models. These capabilities make DESCARTES a unique and powerful tool for researchers in Artificial Intelligence, Cognitive Modelling, and Connectionism.

Acknowledgements

This research has been supported in part by a contract with the JTF program of the DOD and the Office of Naval Research (no. N00014-86-0615). DESCARTES has been implemented on equipment donated to UCLA by Hewlett-Packard, Inc., and Apollo Computer, Inc. We would like to thank John Reeves, Colin Allen, Michael Dyer, and Eduard Hoenkamp for their helpful comments on previous drafts of this paper.

REFERENCES

D'Autrechy, C. L., Reggia, J. A., Sutton, G. G., & Goodall, S. M. (1988): A General-Purpose Simulation Environment For Developing Connectionist Models. *Simulation, 51(1)*, p. 5-19.

Charniak, E. (1986): A Neat Theory of Marker Passing. *Proceedings of the National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA, 1986.

Dyer, M. G. (1989): Symbolic Neuroengineering for Natural Language Processing: A Multi-Level Research Approach. In J. Barnden and J. Pollack, editors, *Advances in Connectionist and Neural Computation Theory*, Ablex Publishing, 1989. In press.

Goddard, N., Lynne, K. J., & Mintz, T. (1986): *Rochestor Connectionist Simulator*. Technical Report TR-233, Department of Computer Science, University of Rochestor.

Hendler, J. (1988): *Integrating Marker-Passing and Problem Solving: A Spreading Activation Approach to Improved Choice in Planning*, Lawrence Erlbaum Associates, Hillsdale, New Jersey.

McClelland, J. L. & Rumelhart, D. E. (1988): *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises*, MIT Press, Cambridge, MA.

Lange, T. & Dyer, M. G. (1989a): Dynamic, Non-Local Role-Bindings and Inferencing in a Localist Network for Natural Language Understanding. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, p. 545-552, Morgan Kaufmann, San Mateo, CA (Collected papers of the IEEE Conference on Neural Information Processing Systems — Natural and Synthetic, Denver, CO, November 1988).

Lange, T. & Dyer, M. G. (1989b): Frame Selection in a Connectionist Model of High-Level Inferencing. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society (CogSci-89)*, Ann Arbor, MI, August 1989.

Lange, T., Hodges J. B., Fuenmayor, M., & Belyaev, L. (1989): *The DESCARTES Users Manual*. Research Report, Computer Science Department, University of California, Los Angeles.

Nenov, V. I., & Dyer, M. G. (1988): DETE: A Connectionist/Symbolic Model of Visual and Verbal Association. *Proceedings of the IEEE Second Annual International Conference on Neural Networks (ICNN-88)*, San Diego, CA, July 1988.

Rumelhart, D. E., & McClelland, J. L. (1986): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Volumes 1-2, MIT Press, Cambridge, MA.

Shastri, L. (1988): A Connectionist Approach to Knowledge Representation and Limited Inference. *Cognitive Science, 12*, p. 331-392.

Vidal, J. & Haggerty, J. (1987): Synchronization in Neural Nets. *Proceedings of the IEEE Conference on Neural Information Processing Systems — Natural and Synthetic (NIPS-87)*, Denver, CO, November 1987.

Waltz, D. & Pollack, J. (1985): Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. *Cognitive Science, 9(1)*, p. 51-74.

Wilson, M. A., Upinder, S. B., Uhley, J.D., & Bower, J. M. (1988): GENESIS: A System for Simulating Neural Networks. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, p. 485-492, Morgan Kaufmann, San Mateo, CA (Collected papers of the IEEE Conference on Neural Information Processing Systems — Natural and Synthetic, Denver, CO, November 1988).

Program of the

Eleventh Annual Conference of the Cognitive Science Society

16-19 August 1989
Ann Arbor, Michigan

LEA LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
1989 Hillsdale, New Jersey Hove and London