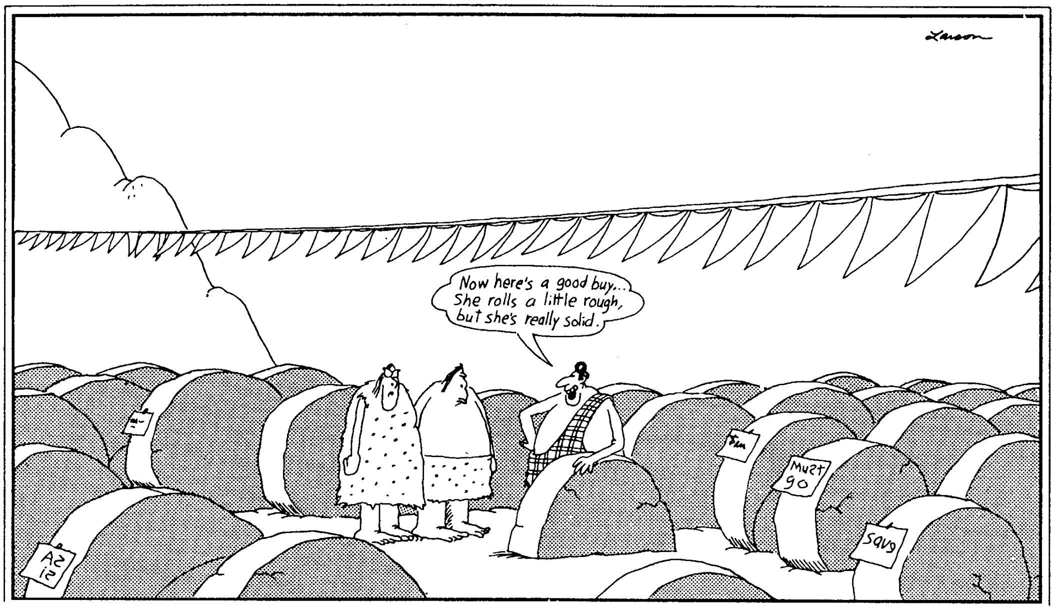
Part I

Representing Mechanical Devices

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

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

○
○
○
○



Early wheeler-dealers

○

○

○

○

Chapter 2
Device Statics

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

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

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

2.1 Physical Objects and Individuals

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

2.1.1 Object Physical Characteristics

In FONM, an object has two types of physical characteristics: its *properties* and its *regions*. Object properties are divided into those related to the object's material composition, which determine how the object can behave under various conditions, and those related to mechanical perturbations, which determine what behavior has happened or can happen. Object properties are represented with physical states. Object regions describe general locations on the object by their shape, and size, and determine how and where the object can be connected and manipulated.

Physical States

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

Properties

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

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

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

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

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

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

Dimension and Direction

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

	X/Z	Y	Z	R
translation	along-length	along-width	along-depth	along-radius
rotation	about-length	about-width	about-depth	about-radius

Table 2.1 : Symbolic representations for the six cartesian coordinate degrees of freedom.

Each of the entries in Table 2.1 represents a motion *degree of freedom*. References to these dimensions will be used to represent behavior enablement and disablement. For example, when a restraint state is represented, the dimension (and direction) specify degrees of freedom which are fixed with respect to motion. When a force state is represented, the dimension (and direction) specify the degrees of freedom which, if unfixed, will result in motion, and, if fixed, will result in force transmission.

What is normally termed the theta dimension is captured by the ABOUT-LENGTH dimension.

Direction is described using a dimension and value. The value can be POS or NEG, with respect to the object's center of gravity (CG), or it can be an object or object region. A POS value indicates the direction toward the object CG, while a NEG value indicates the direction away from the object CG, (see Fig. 2.1a,b).

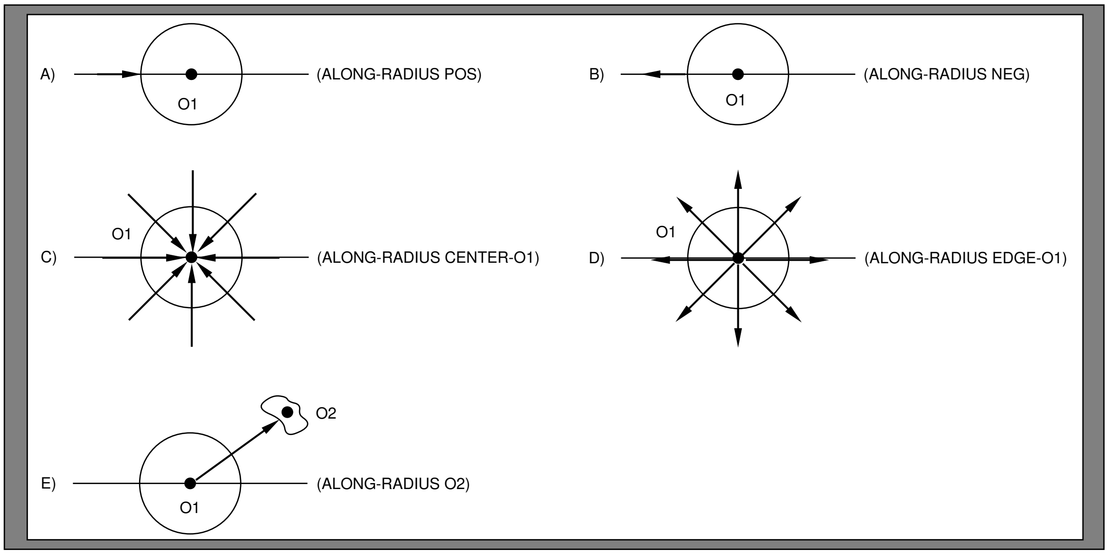


Figure 2.1 Direction in FONM. Examples A and B use a dimension and value, and examples C, D and E use a dimension and an object or region.

In Fig. 2.1, six ways to represent direction with radial dimensions are illustrated. O1 represents a circular object and the arrows represent directions. The same mechanisms are used to represent direction for translational dimensions. Consider the spoon illustrated in

Fig. 2.2

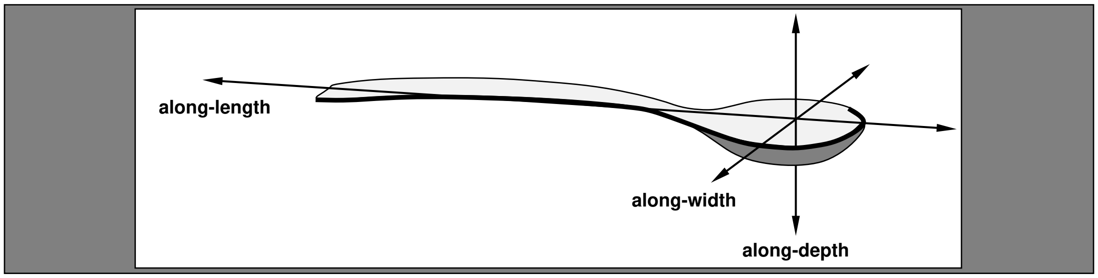


Figure 2.2 Cartesian dimension reference for a spoon.

Using the notational convention for representing direction illustrated in Fig. 2.1a, b, an applied restraint on a spoon handle, caused by a person holding the spoon, can be represented (using slot-filler notation) as follows:

```
(state REST-SP1
  obj      SPOON.1
  prop     RESTRAINT
  dimr     ((ALONG-DEPTH NEG))))
```

Figure 2.3 Representation example for restraint states.

The dimr role is bound to ((ALONG-DEPTH NEG)), which describes a restraint applied away from the spoon handle CG in the spoon's thickness dimension (i.e., cartesian z or global downward). Note that there is no value for the restraint state represented. The restraint dimension and direction are equivalent to the state value.

The representation of direction using a dimension and a reference location or object, as depicted in Fig. 2.1c-e, is shown in Fig. 2.4. Consider a soup spoon being lowered into some soup. The spoon is propelled toward the soup, which causes an applied force state on the spoon handle and enables motion.

```
(state FORCE-SP1
   obj          SPOON.1
   prop         AFORCE
   enables      (process MOTION-LINEAR-SP1
                   dimr      ((ALONG-DEPTH STUFF-SOUP))))
```

Figure 2.4 Representation example for explicit direction.

The force FORCE-SP1 enables spoon motion, represented by the process MOTION-LINEAR-SP1. The motion dimr role is bound to ((ALONG-DEPTH STUFF-SOUP)), which means 'toward the soup.' In this example, the dimension of motion is simply described because it coincides with a spoon axis. In the more general case, however, the dimension may be more difficult to describe, since motion toward the soup might not be along a know object axis. A representation which alleviates this problem references the direction but ignores the dimension:

```
(state FORCE-SP1
   obj         SPOON.1
   prop        AFORCE
   enables     (process MOTION-LINEAR-SP1
                  dimr      ((STUFF-SOUP)))))
```

Figure 2.5 Representation example for implicit direction.

When a dimension is represented with no direction, both directions are assumed. In the representation of SPOON.1 length below, ((ALONG-LENGTH)) describes both directions of the spoon's longitudinal axis:

```
(state SIZE-AL-SP1
   obj         SPOON.1
   prop        SIZE
   dimr        ((ALONG-LENGTH))
   val         QUANT.1)
```

Figure 2.6 Representation for entire dimensions.

Quantity Values and Quantity Spaces

The state *val* slot refers to a quantity and is filled by a knowledge structure called quant. Quant has three roles: state, unit, and val. The *unit* role is used to distinguish between and enable translations between quantities. The unit filler always takes on the symbolic name for the value being described. The quant *val* slot can be represented with a quantitative value, as a point in a one-dimensional quantity space. A *quantity space* is a continuum of values associated with a specific unit type, such as inches or pounds. Each point on the quantity space represents a value of an object property. The quantity space is bound by the maxima and minima of known values.

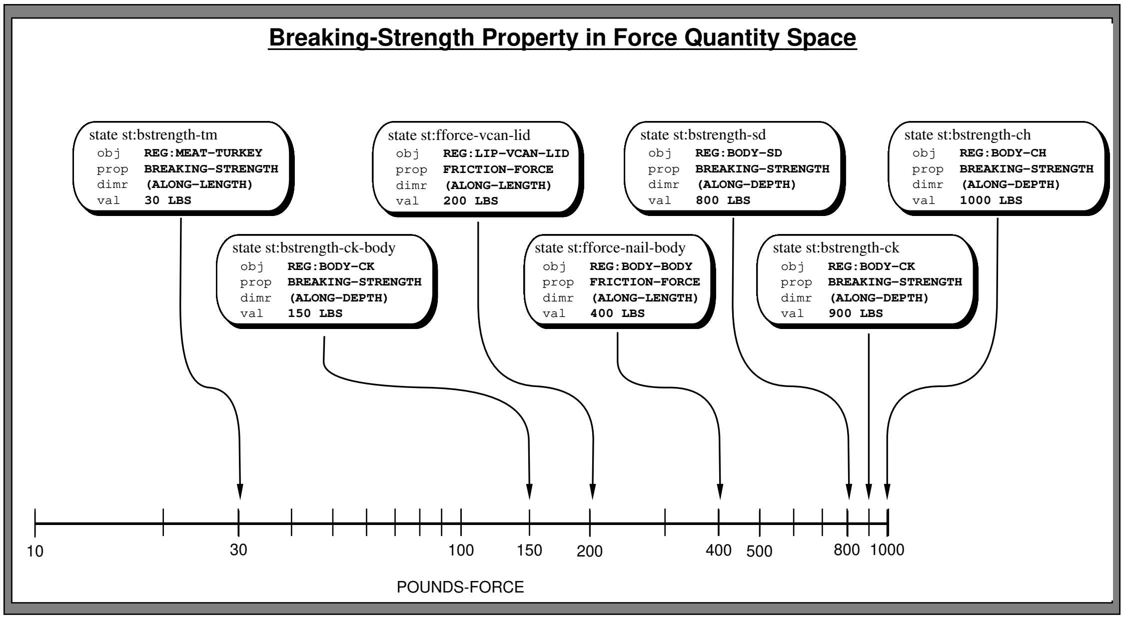


Figure 2.7 A portion of the quantity space for force, measured in pounds.

Fig. 2.7 illustrates a one-dimensional quantity space for pounds-force (a logarithmic scale) and a number of object states plotted on the scale. Object property values, represented as quantities, are related to other objects, and to other quantities, on a quantity space. The breaking strengths for three devices (screwdriver, a claw hammer, and a carving knife) and some turkey meat are plotted above on the pounds-force quantity space. Two friction force values (the friction force between a varnish can lid and its vessel, and a nail and its medium) are also represented on the line.

The quantity space enables a number of useful comparisons. For example, the breaking strengths of the devices plotted can be compared. Second, this quantity space provides a means for comparing breaking strengths and friction forces through the common unit of measure. Finally, a quantity space also provides a simple way to compare values in different units of measurement, such as pounds and kilograms, by their common objects.

A quantity value can also be represented qualitatively, using a simple (<NOM, NOM, >NOM) scale. In this alternate case, the *unit* role is empty. For example, a significant characteristic of a broken ruler is its change in size (shorter). Without knowing the exact size change, the new size can be represented qualitatively as shown in Fig. 2.8:

```
(state SIZE-AL-RUL1
  obj      RULER.1
  prop     SIZE
  dimr     ((ALONG-LENGTH))
  val      (quant
              state    SIZE-AL-RUL1
              unit     NIL
              val      <NOM)))
```

Figure 2.8 Representation example for quantity.

In this case, there is no way to determine what the new length of the ruler is with respect to other devices, but it may be suitable for function-related inferences, such as the ruler not being suitable for precise measurement. It should be noted that this representation doesn't make mention of which piece of the ruler is being represented, only that the ruler is no longer its original size.

Object Regions

In FONM, appearance is represented with a knowledge structure called a region. Regions describe general areas or volumes of an individual object by their shape, size, and location. Regions are used to represent objects having complex shapes.

Regions are used to describe objects having locations which have something to do with their function and behavior, but which may not require specific reference in a particular coordinate system. For example, consider a table top. In FONM, the table top is represented as a single flat surface, and all points on it can be applied equally well to the support-related tasks that any specific point can. In this respect, the region is a general location specifier for behavior, and the number of points that need to be represented in order to make inferences about table top behavior is dramatically reduced. Now consider a gear tooth. The tooth may be a complex, repetitive, shape but that shape can be described simply as a convex surface on the edge of a wheel. The fact that it is repetitive allows us to ignore the other teeth, since they all can be used to accomplish the same tasks. Stating that the gear tooth is located on the edge, but not where on the edge, does not detract from the ability to say what behavior the tooth can engage in, or what the function of the gear will be.

A region knowledge structure is defined by four roles: obj, shape, size, and loc. The *obj* role references the object in question. The *shape* role is associated with the curvature of the region. The *size* role is used as a comparison metric with other regions. The *loc* role is used to describe the general vicinity of the region with respect to the object's center of gravity or to other regions.

Shape

A region's shape is its most apparent feature. In FONM, regions are described as surfaces or surface intersections, and shape is represented as a the curvature of the surface or intersection. A *surface* region describes an affected volume. There are three types of surface regions defined by their general curvature: concave surfaces are called *indentations*; and are associated with containment volumes, flat surfaces are called *faces*; and are associated with support areas or volumes, and convex surfaces are called *protuberances*, and are associated with path interference and confinement. A tree illustrating the region classes used in

FONM, with respect to specialization along these metrics, is shown in Fig. 2.9

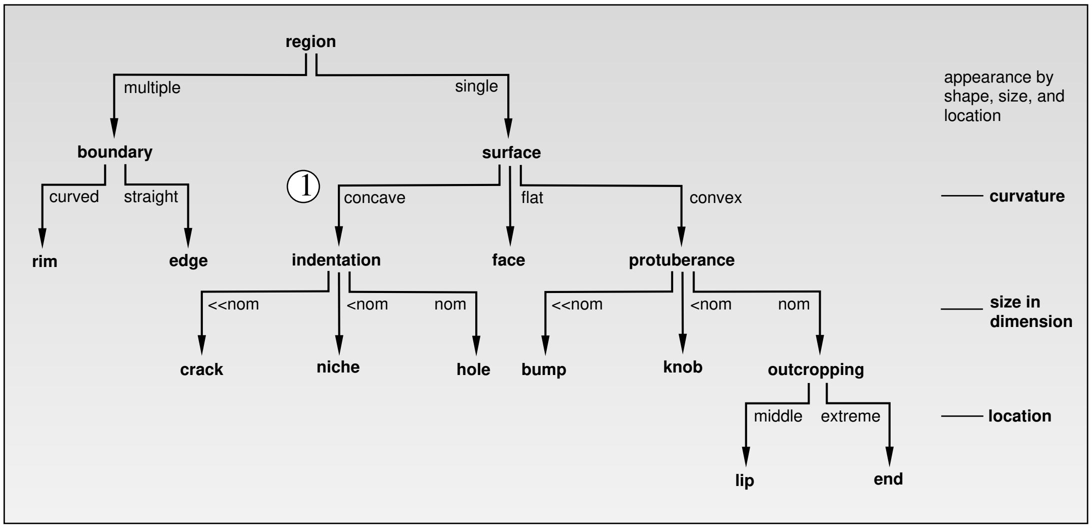


Figure 2.9 Region hierarchy in FONM.

The items along the right hand side of Fig. 2.9 describe the different characteristics of a region (i.e., curvature, size, and location). For each of these characteristics, the tree is differentiated by specializing along a particular value. For example, at branch (1) the curvature is concave and the subtree is associated with indentations.

Surface intersections (or junctions) are called *boundaries*. In Fig. 2.9, an edge is represented as a straight boundary (e.g., between two faces), and a rim is represented as a curved boundary (e.g., between an indentation and a protuberance).

Location

A region's general location is described with respect to some known location on the object. A location is represented with a knowledge structure called *location*. Locations have two roles: ref and dimv. The *ref* role refers to a reference region or to the object's center of gravity. The *dimv* role is used to locate the region, with respect to the reference, with dimension/value pairs. The values used in FONM are the discrete set: FAR-LEFT, LEFT, NEAR-LEFT, MIDDLE, NEAR-RIGHT, RIGHT, and FAR-RIGHT. Using this notation, the eraser end of a pencil is represented as a convex region located along the length of the pencil at its extreme value:

```
(region PR-END1-P1
   obj        PENCIL.1
   shape      CONVEX-BLUNT
   loc        (location
                 dimv     ((ALONG-LENGTH FAR-RIGHT))))
```

Figure 2.10 Representation example for region location values.

In this example, the region's size is ignored, and the ref role is empty because the value refers to the object CG (all regions). In FONM, no distinction is made between the extrema, such as FAR-LEFT and FAR-RIGHT, unless another region is represented. In the pencil case, there is another end, the pointed end:

```
(region PR-END2-P1
   obj         PENCIL.1
   shape       CONVEX-POINTED
   loc         (location
                  dimv     ((ALONG-LENGTH FAR-LEFT))))
```

Figure 2.11 Representation example for an end region.

which now enables a distinction with respect to the other end region's location. The location notation follows the same convention as does dimension/direction for state descriptions.

A pencil has a third region called a handle. The handle location can be anywhere along the length of the pencil, since a person could hold the pencil anywhere along its length equally well.

```
(region PR-HANDLE-P1
   obj         PENCIL.1
   shape       CONVEX
   loc         (location
                  dimv     ((ALONG-LENGTH))))
```

Figure 2.12 Representation example for a body region.

If a dimension is referenced with no value, as with the pencil handle above, then the entire dimension is assumed.

Size

Regions are also distinguished by their size, but the values for size used in this context refer to the region type rather than a unit type. The baseline assumption is that the region's size is the same in all dimensions, so the size state value is based on selecting the dimension or dimensions which are used as a comparison. The first dimension given is the reference size and the others, if given, as values, refer to it. For example, a crack is an indentation which is very long with respect to its width. The size of a crack is represented with respect to its width, so the crack size in the ALONG-WIDTH dimension is the reference. The size value of the crack in the other dimensions is then taken relative to the crack's width: the size along its length is much greater than nominal (>>NOM) and the size along its depth is greater than nominal (>NOM).

```
(state CR-CM1
  obj      IND-CR-CM1
  prop     SIZE
  dimr     ((ALONG-WIDTH))
  val      ((ALONG-LENGTH >>NOM)
            (ALONG-DEPTH  >NOM)))
```

Figure 2.13 Representation example for a crack indentation size.

2.2 Object Types

2.2.1 Linkages and Stuff

There are two classes of individual objects in FONM: *linkages* and *stuff*. Linkages make up the components of devices, and stuff is everything else. Linkages and stuff are differentiated by their material stiffness properties. A *linkage* is an elastic object; meaning that it is made of a material capable of transmitting mechanical force in at least one direction along at least one of its dimensional axes. Most mechanical objects are linkages, because they can transmit force along entire dimensions. For example, a pipe is a linkage in all dimensions, and directions. Wires, cables, and ropes are examples of linkage objects in a single dimension/direction, because they can transmit force under tension but not under compression.

Objects are represented with a phys-obj knowledge structure which is associated with an object type and its physical characteristics. Consider the representation of a linkage object shown below.

```
(phys-obj LINKAGE-OBJECT
   has-phys     (MS-LINKAGE-STIFFNESS
                 LINKAGE-END1
                 LINKAGE-END2))

(state MS-LINKAGE-STIFFNESS
   obj         LINKAGE-OBJECT
   prop        MATERIAL-STIFFNESS
   val         (quant ELASTIC-MS
                  units    PPI
                  val      >NOM))

(region LINKAGE-END1
   loc         (location LOC1-LINKAGE
                  dimv     ((?DIM      FAR-LEFT))))

(region LINKAGE-END2
   loc         (location LOC2-LINKAGE
                  dimv     ((?DIM      FAR-RIGHT))))
```

Figure 2.14 Representation example for a linkage object.

Since the linkage object is defined by its ability to transmit force, only its material stiffness and the regions where it can transmit the force are included in the FONM representation. The unit for stiffness shown in the representation for `MS-LINKAGE-STIFFNESS` is pounds-per-inch, PPI. The nominal stiffness value is somewhat arbitrary, and is intended to represent the minimum stiffness which would preclude an object from being able to transmit mechanical force.

Stuff is used to represent objects which cannot transmit mechanical forces in any dimension or direction. Examples of objects which can be represented with stuff are liquids and gases, since they require a container (such as a piston and cylinder) to effect mechanical force transmission. Stuff may become a linkage through state change (e.g., thermodynamic), however, in FONM, non-mechanical effects on objects are not represented.

2.2.2 Physical Type and Geometric Object Primitives

In FONM, every linkage object has a physical type based on one of ten idealized geometric objects: cylinder, column, filament, cone, sphere, block, plate, film, wedge, and pyramid. Each of these objects is defined by its (a) regions, (c) center of gravity, (d) size, and (e) shape, as illustrated in Fig. 2.15. Each object also has associated with it a set of dimensions for referring to its size and shape.

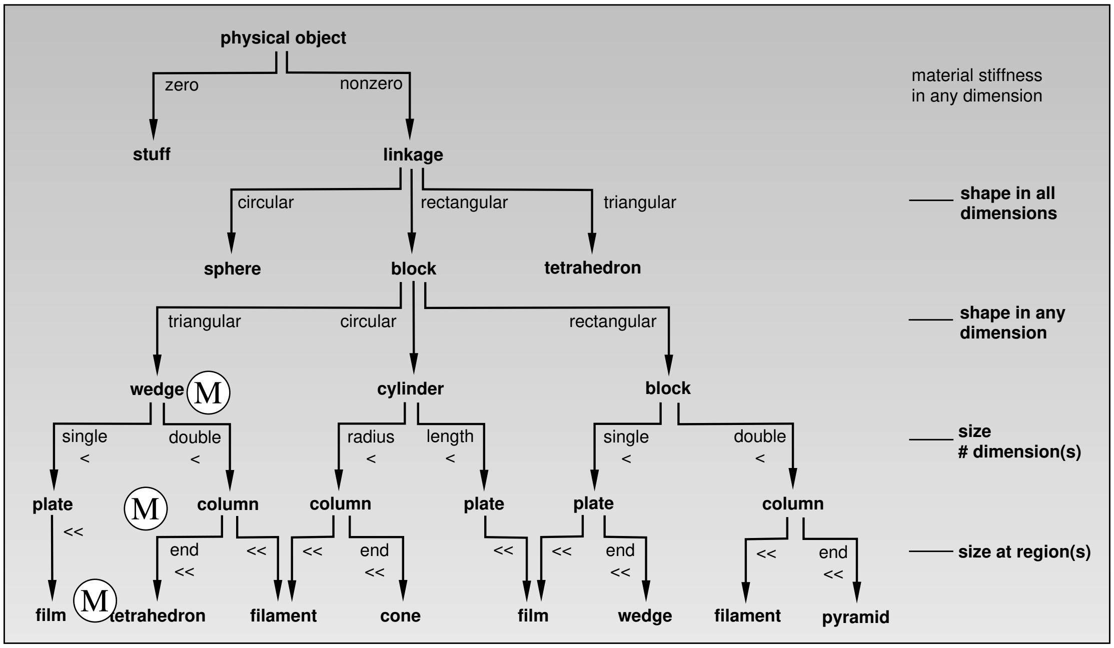


Figure 2.15 Geometric object hierarchy in FONM.

Fig. 2.15 is organized by shape and size specializations, similar to that of regions. Objects which have the same shape in every dimension (circular => "sphere", rectangular => "block", and triangular => "tetrahedron") are shown in the first tier. The "block" sub-hierarchy is shown expanded. It should be noted that this figure does not represent a unique tree. There are different ways to combine these specializations and come up with the same categories. For example, a tetrahedron can be described as a linkage object which is triangular in all dimensional views. A tetrahedron can also be described as a rectangular block that is first modified to be triangular in one dimension (a wedge), then squished and lengthened into a column, and then sharpened at one end. This path is labeled with (M)s. Note the repetitive nature of shapes in Fig. 2.15. Every shape leads to a plate and column, as well as to other standard shapes. This suggests that the shapes chosen in the first tier are acceptable geometric primitives as long as size modifications are allowed. The geometric primitives chosen for FONM include size modifications.

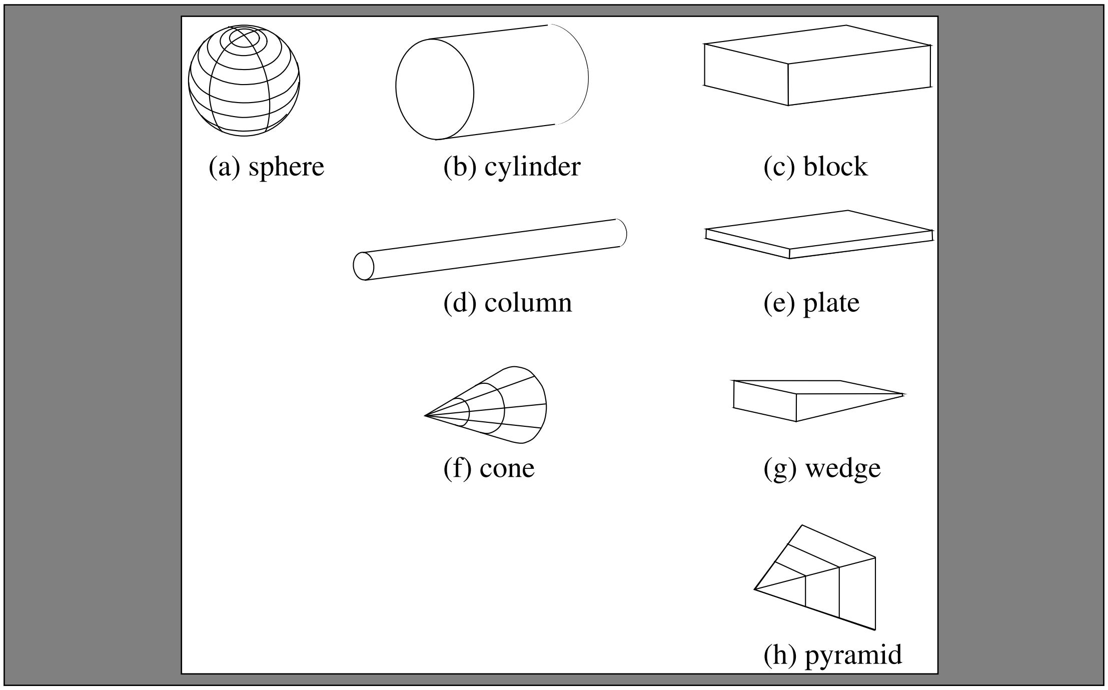


Figure 2.16 Geometric object primitives in FONM. Sphere, cylinder, column, cone, block, plate, wedge, and pyramid.

2.2.1 Cylinder

The cylinder is a fundamental geometric entity. An idealized cylinder is depicted in Fig. 2.17.

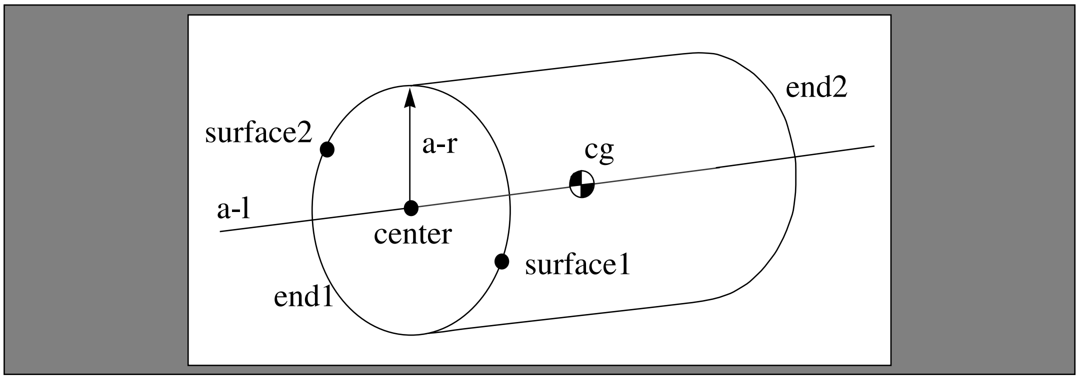


Figure 2.17 An idealized cylinder.

Cylinder Coordinate Axes

The cylinder is described with four dimensions: along-length (a-l), about-length (ab-l), along-radius (a-r), and about-radius (ab-r).

Cylinder Center of Gravity

Each object has a center of gravity (CG), which represents the location of the object in the absence of any other reference to object location. Center of gravity is not considered a region in FONM, because it is not associated with appearance. The cylinder CG is located at the mid-point on both translational dimensions (i.e., a-l middle a-r middle).

Cylinder Regions

The cylinder has five regions: a center [of rotation], two ends, and two surface points. The regions are located with respect to the object center of gravity. That is, the mid-points of the dimensions are coincident with the CG.

Name	Dimension	Value
center	along-radius	middle
end1	along-length	far-left
end2	along-length	far-right
surface1	along-radius	far-right
surface2	along-radius	far-left

Table 2.2 Cylinder Regions

Surface1 and surface2 describe the outer surface of the cylinder as the endpoints of any particular diameter. Thus, surface1 and surface2 are position (ALONG-LENGTH) independent. The dimension and value entries in Table 2.2 represent the elements of the dimv location role, and the values are same as the symbolic notation presented for region location (e.g., MIDDLE).

Cylinder Size

The size of the cylinder is arbitrary, except for two requirements. First, the radial size is the same at all locations along the cylinder length. Second, the length and radial sizes are related as follows:

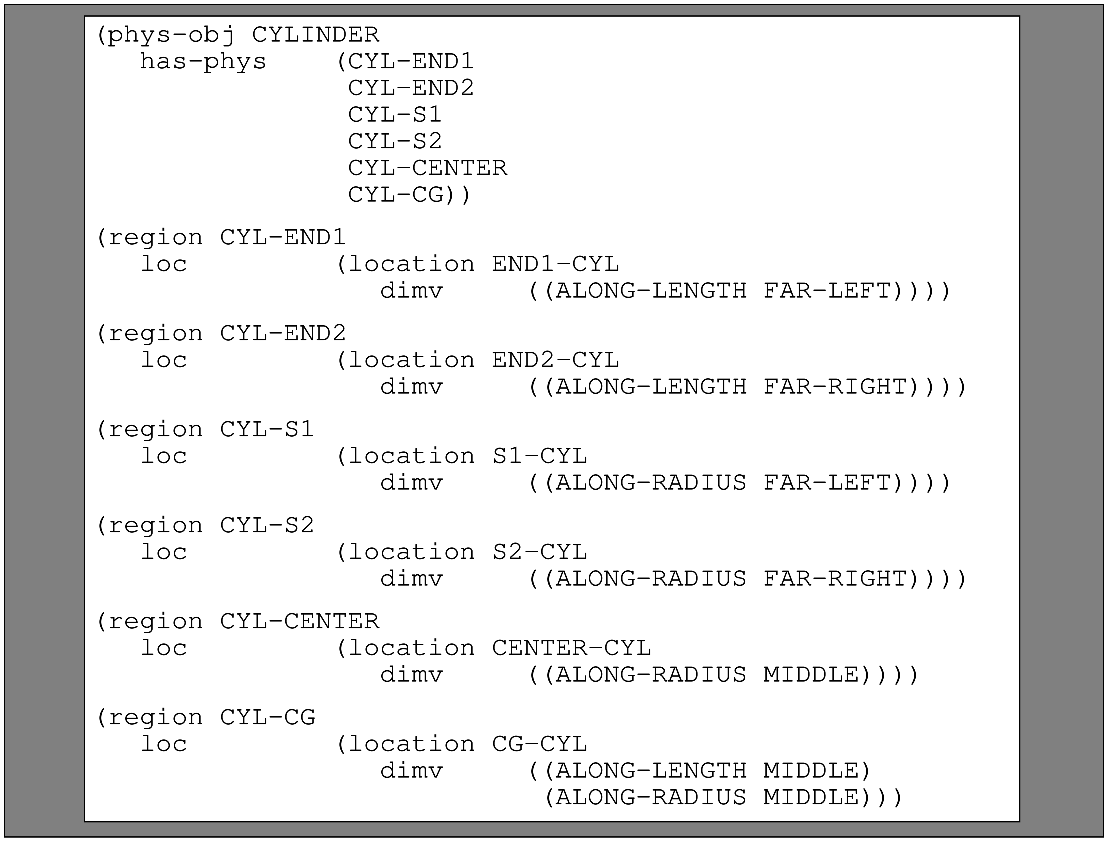
$$\text{(size along-length)} > \text{(size along-radius)} \tag{2-1}$$

where the greater-than sign is a symbolic representation that means the size in one dimension is greater, but within an order of magnitude, of the other:

$$\begin{aligned}&\text{(size along-radius)} > 1/10 \text{ (size along-length) and}\\&\text{(size along-radius)} < \text{(size along-length)}\end{aligned} \tag{2-2}$$

Cylinder Representation

The regions representations for the cylinder geometric object are shown below.



```
(phys-obj CYLINDER
   has-phys    (CYL-END1
                CYL-END2
                CYL-S1
                CYL-S2
                CYL-CENTER
                CYL-CG))

(region CYL-END1
   loc         (location END1-CYL
                  dimv     ((ALONG-LENGTH FAR-LEFT))))

(region CYL-END2
   loc         (location END2-CYL
                  dimv     ((ALONG-LENGTH FAR-RIGHT))))

(region CYL-S1
   loc         (location S1-CYL
                  dimv     ((ALONG-RADIUS FAR-LEFT))))

(region CYL-S2
   loc         (location S2-CYL
                  dimv     ((ALONG-RADIUS FAR-RIGHT))))

(region CYL-CENTER
   loc         (location CENTER-CYL
                  dimv     ((ALONG-RADIUS MIDDLE))))

(region CYL-CG
   loc         (location CG-CYL
                  dimv     ((ALONG-LENGTH MIDDLE)
                            (ALONG-RADIUS MIDDLE)))
```

Figure 2.18 Representation for cylinder geometric primitive.

The size characteristics of a cylinder geometric primitive can be specialized to represent three other geometric primitives: (1) circular plate, (2) column, and (3) cone. The plate and column (and filament) represent objects which have size specializations which are effected at both end regions of the cylinder, while the cone represents an object where the size specialization occurs at one region. A plate results from having a size in the along-radius dimension being much larger than that in the along-depth dimension. The column is the opposite case, where the size in the along-depth dimension is much larger than that in the along-radius dimension. The filament is simply an extreme case of a column. A cone, described below, is a cylinder specialization where the radius at one end is much smaller than that at the other end. This same specialization can be applied to a column to produce a point.

2.2.2 Cone

The idealized cone is defined as a cylinder where the radius at one end region is zero, as shown below (Fig. 2.19).

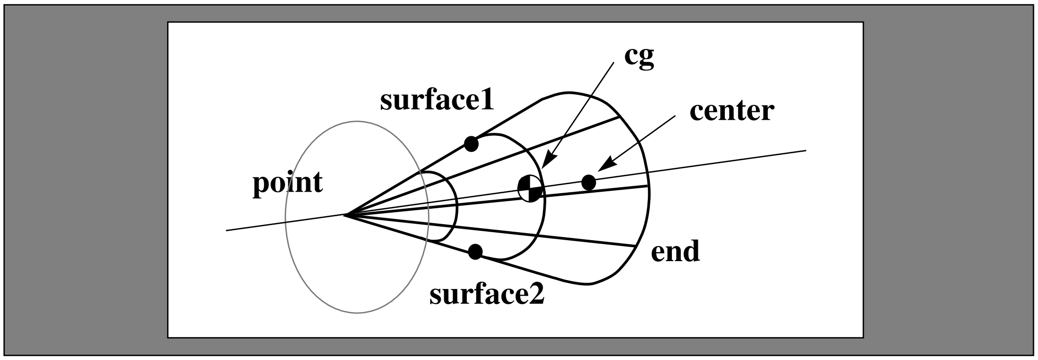


Figure 2.19 Idealized cone and regions.

The dotted line at the point end of the cone illustrates the end where the cylinder/column/filament radius has been reduced. As a result, the surface regions (surface1 and surface2) have different radii along the length of the object, and the CG shifts towards the end.

Cone Axes and Size

The dimensions used to define a cone, and the sizes of a cone, are identical to those for a cylinder.

Cone Center of Gravity

The cone CG is located at the mid-point along the cone radius (a-r) and one-third the distance from its end (2/3 from its point).

Cone Regions

The cone has five regions differing slightly from those of a cylinder: a center [of rotation], a point, an end, and two surface points. The region locations, with respect to the cone CG, are shown in Table 2.3

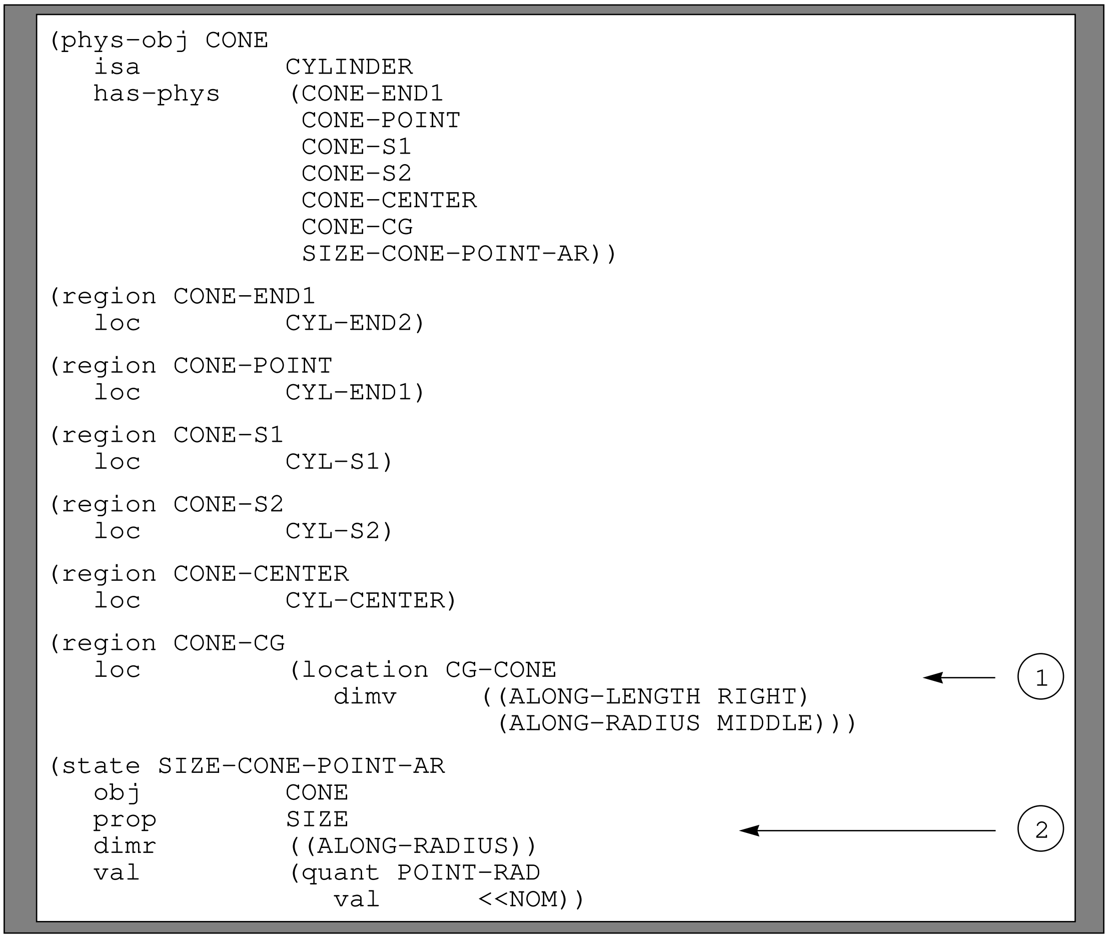
Name	Dimension	Value
center	along-radius	middle
point	along-length along-radius	far-left middle
end	along-length	far-right
surface1	along-radius	far-right
surface2	along-radius	far-left

Table 2.3 Cone Regions

Surface1 and surface2 describe endpoints of any particular diameter and change as the radius changes along the length of the cone.

Cone Representation

The cone representation is illustrated with respect to its differences from the cylinder At (1), the CG location has a symbolic location at the RIGHT value, which is somewhat arbitrary and intended to represent the shifting toward the end rather than be the true location of the CG. At (2), the radial size of the point region is represented symbolically, with respect to the object radial dimension, as <<NOM.



```
(phys-obj CONE
   isa         CYLINDER
   has-phys    (CONE-END1
                CONE-POINT
                CONE-S1
                CONE-S2
                CONE-CENTER
                CONE-CG
                SIZE-CONE-POINT-AR))

(region CONE-END1
   loc         CYL-END2)

(region CONE-POINT
   loc         CYL-END1)

(region CONE-S1
   loc         CYL-S1)

(region CONE-S2
   loc         CYL-S2)

(region CONE-CENTER
   loc         CYL-CENTER)

(region CONE-CG
   loc         (location CG-CONE
                  dimv     ((ALONG-LENGTH RIGHT)
                            (ALONG-RADIUS MIDDLE)))

(state SIZE-CONE-POINT-AR
   obj         CONE
   prop        SIZE
   dimr        ((ALONG-RADIUS))
   val         (quant POINT-RAD
                  val      <<NOM))
```

Figure 2.20 Representation for cone geometric primitive.

2.2.3 Sphere

A sphere is a fundamental geometric shape that is circular in every dimensional view. An idealized sphere is illustrated in Figure 2.21

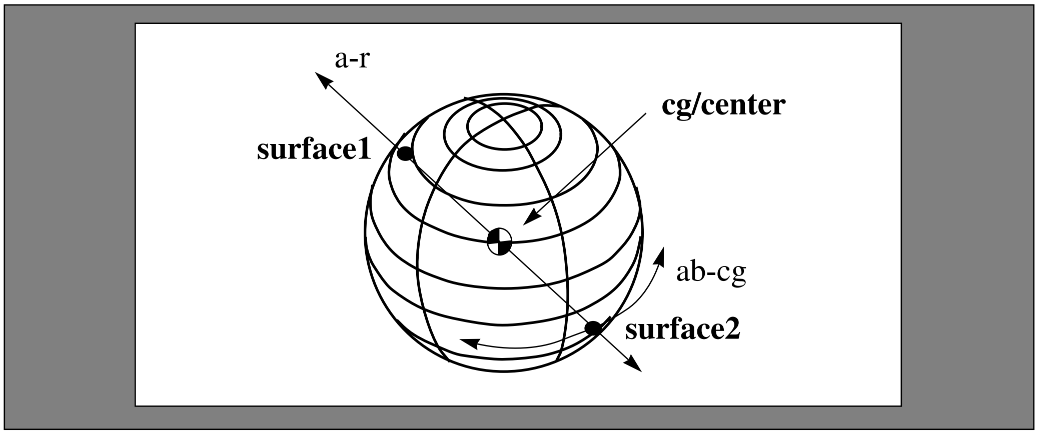


Figure 2.21 Idealized sphere and regions.

Sphere Coordinate Axes

The sphere has three dimensional axes, along-radius (a-r), about-radius (ab-r), and about-cg (ab-cg).

Sphere Center of Gravity

The sphere CG is located at the mid-point along the sphere radius, and coincides with the center region.

Sphere Regions

The sphere has three regions: a center [of rotation] and two surface points. The regions are located, with respect to the sphere CG, as shown in Table 2.4. Surface1 and surface2 describe endpoints of any particular diameter.

Name	Dimension	Value
center	along-radius	middle
surface1	along-radius	far-right
surface2	along-radius	far-left

Table 2.4 Sphere Regions

Sphere Size

There are no size restrictions on sphere objects.

Sphere Representation

```
(phys-obj SPHERE
   has-phys    (SPHERE-S1
                SPHERE-S2
                SPHERE-CENTER
                SPHERE-CG
                SHAPE-SPHERE))

(region SPHERE-S1
   loc         (location S1-SPHERE
                  dimv     ((ALONG-RADIUS FAR-LEFT))))

(region SPHERE-S2
   loc         (location S2-SPHERE
                  dimv     ((ALONG-RADIUS FAR-RIGHT))))

(region SPHERE-CENTER
   loc         (location CENTER-SPHERE
                  dimv     ((ALONG-RADIUS MIDDLE))))

(region SPHERE-CG
   loc         SPHERE-CENTER)

(state SHAPE-SPHERE
   obj         SPHERE
   prop        SHAPE
   val         CIRCULAR)
```

Figure 2.22 Representation of sphere geometric primitive.

2.2.4 Block

A block is a fundamental geometric shape which is rectangular in all dimensional views. An idealized block is any rectangular solid with planar faces.A rectangular block is illustrated in Figure 2.23, however, a block can be any parallelepiped or trapezoid as well.

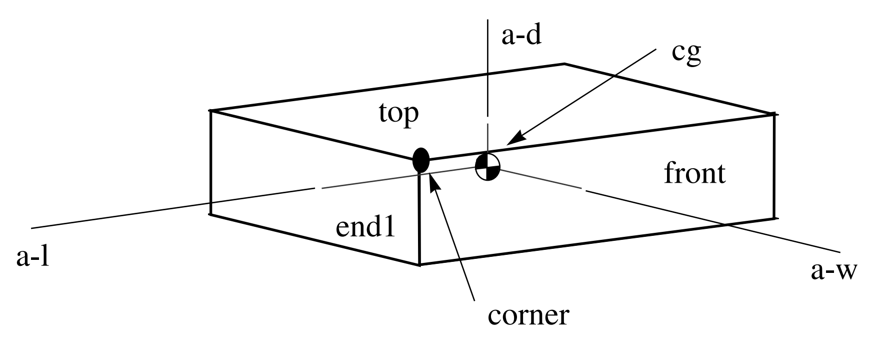


Figure 2.23 Idealized block and some face regions (top, front, and end1) labeled. A corner region is also labeled.

Block Coordinate Axes

The block has three dimensional axes, along-length (a-l), along-width (a-w), and along-depth (a-d).

Block Center of Gravity

The block CG is located at the intersection of mid-points along all parallel face translational axes, and at the mid-point between non-parallel face translational axes. In the right-angled block, the CG is located at (a-l middle a-w middle a-d middle).

Block Regions

A block has six face, twelve edge, and eight corner regions. The face region locations are defined in Table 2.5.

Face Name	Dimension	Value
top	along-depth	far-left
bottom	along-depth	far-right
front	along-width	far-left
back	along-width	far-right
end1	along-length	far-left
end2	along-length	far-right

Table 2.5 Block Regions

Edge regions are defined as boundaries of two faces, such as front and top. Corners are represented as protuberances, which in this case are boundaries of three faces, such as front, end1, and top.

Block Size

The block size restrictions similar to that of a cylinder. All three sizes are approximately the same:

$$\text{(size along-length)} = \text{(size along-width)} = \text{(size along-depth)} \tag{2-3}$$

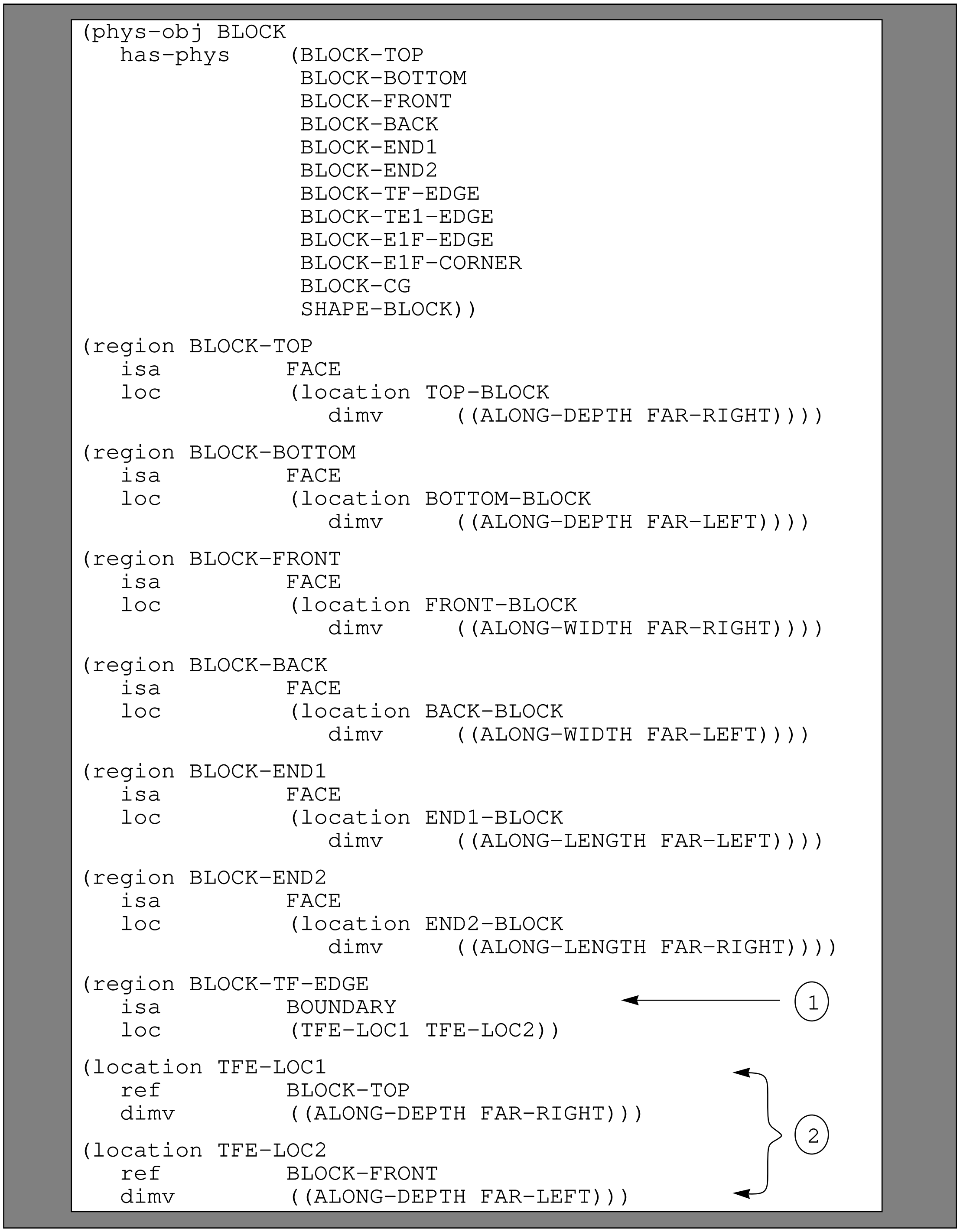
where the equals sign means approximately the same size, and, if anything, the size in length and width is greater than that in depth, but not as much as an order of magnitude.

Block Shape

The block is quadrilateral (rectangular, parallelogram, or trapezoidal) in all dimensions.

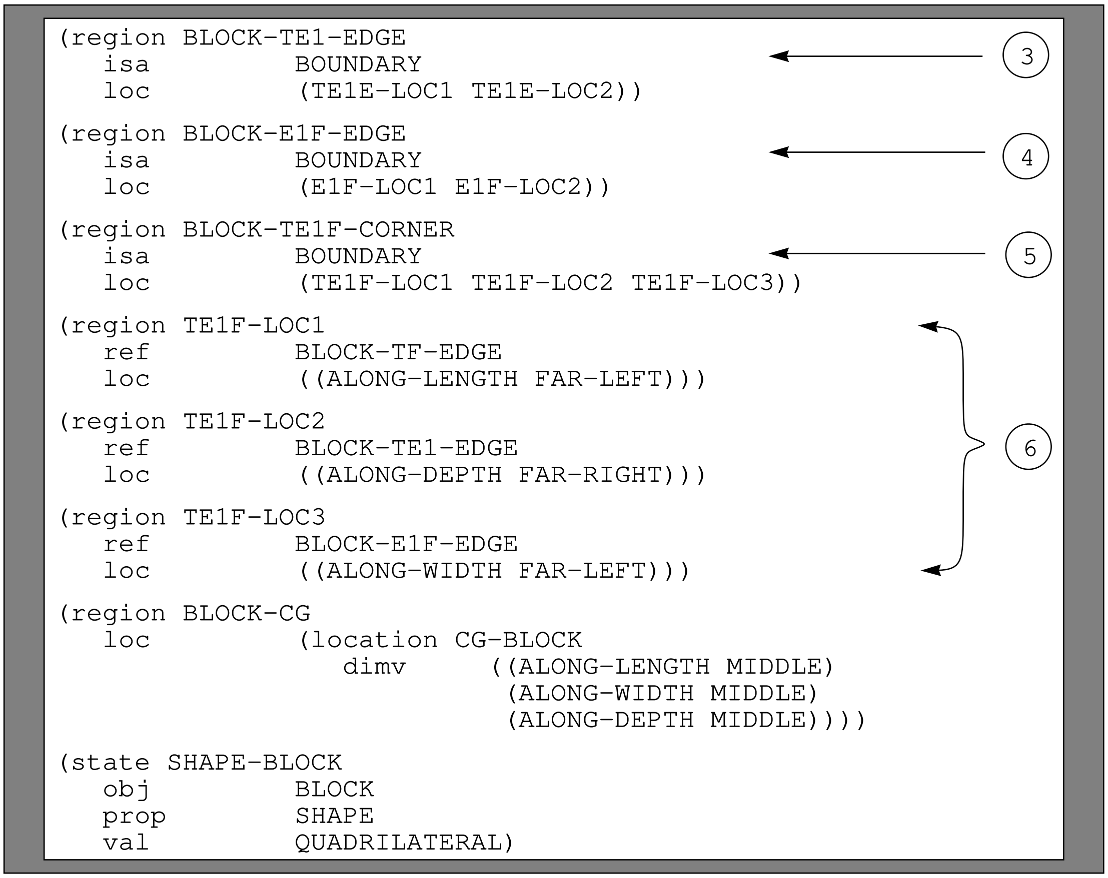
Block Representation

The representation for a block is large owing to the number of edges and corners. A portion of the block representation is presented below, with all faces, three edges, and a corner. The faces are represented as shown in Table 2.5. The edges are represented as boundary locations on faces. For example, at (1) and (3), below, the top-front edge, and the top-end



```
(phys-obj BLOCK
   has-phys    (BLOCK-TOP
                BLOCK-BOTTOM
                BLOCK-FRONT
                BLOCK-BACK
                BLOCK-END1
                BLOCK-END2
                BLOCK-TF-EDGE
                BLOCK-TE1-EDGE
                BLOCK-E1F-EDGE
                BLOCK-E1F-CORNER
                BLOCK-CG
                SHAPE-BLOCK))

(region BLOCK-TOP
   isa         FACE
   loc         (location TOP-BLOCK
                  dimv     ((ALONG-DEPTH FAR-RIGHT))))

(region BLOCK-BOTTOM
   isa         FACE
   loc         (location BOTTOM-BLOCK
                  dimv     ((ALONG-DEPTH FAR-LEFT))))

(region BLOCK-FRONT
   isa         FACE
   loc         (location FRONT-BLOCK
                  dimv     ((ALONG-WIDTH FAR-RIGHT))))

(region BLOCK-BACK
   isa         FACE
   loc         (location BACK-BLOCK
                  dimv     ((ALONG-WIDTH FAR-LEFT))))

(region BLOCK-END1
   isa         FACE
   loc         (location END1-BLOCK
                  dimv     ((ALONG-LENGTH FAR-LEFT))))

(region BLOCK-END2
   isa         FACE
   loc         (location END2-BLOCK
                  dimv     ((ALONG-LENGTH FAR-RIGHT))))

(region BLOCK-TF-EDGE
   isa         BOUNDARY
   loc         (TFE-LOC1 TFE-LOC2))

(location TFE-LOC1
   ref         BLOCK-TOP
   dimv        ((ALONG-DEPTH FAR-RIGHT)))

(location TFE-LOC2
   ref         BLOCK-FRONT
   dimv        ((ALONG-DEPTH FAR-LEFT)))
```

Figure 2.24 Representation for block geometric primitive (plate 1).



```
(region BLOCK-TE1-EDGE
   isa          BOUNDARY
   loc          (TE1E-LOC1 TE1E-LOC2))

(region BLOCK-E1F-EDGE
   isa          BOUNDARY
   loc          (E1F-LOC1 E1F-LOC2))

(region BLOCK-TE1F-CORNER
   isa          BOUNDARY
   loc          (TE1F-LOC1 TE1F-LOC2 TE1F-LOC3))

(region TE1F-LOC1
   ref          BLOCK-TF-EDGE
   loc          ((ALONG-LENGTH FAR-LEFT)))

(region TE1F-LOC2
   ref          BLOCK-TE1-EDGE
   loc          ((ALONG-DEPTH FAR-RIGHT)))

(region TE1F-LOC3
   ref          BLOCK-E1F-EDGE
   loc          ((ALONG-WIDTH FAR-LEFT)))

(region BLOCK-CG
   loc          (location CG-BLOCK
                   dimv      ((ALONG-LENGTH MIDDLE)
                              (ALONG-WIDTH MIDDLE)
                              (ALONG-DEPTH MIDDLE))))

(state SHAPE-BLOCK
   obj          BLOCK
   prop         SHAPE
   val          QUADRILATERAL)
```

Figure 2.25 Representation for block geometric primitive (plate 2).

edge are represented. At (2), the top-front edge is located with respect to both the top and front faces. The locations of the top-end1 and end1-front edges are similarly represented. The top-end1-front corner (5) is represented as the boundary of the edges just described, and the locations are represented at (6). Other corners are represented in a similar manner.

Only one edge and one corner have been shown in this representation. The other edges and corners are represented in the same fashion. Note that edges require references to each face, and that corners are boundary intersections just as boundaries are surface intersections.

Like the cylinder, the block can be specialized with respect to size and location to describe four other geometric primitives: (1) square plate, (2) column, (3) wedge, and (4) pyramid. The plate and column (and filament) represent objects which have size specializations which are effected at both opposing face regions of the block, while the while the wedge and pyramid represent objects where the size specialization occurs at one face region. The pyramid extrapolates the size change from one dimension to both dimensions at the region. A plate results from having a size in the along-length and along-width dimensions being approximately the equal but much larger than that in the along-depth dimension. The column is the opposite case, where the size in the along-depth dimension is much larger than that in the along-length and along-width dimensions (but they are still roughly equivalent). The filament is simply an extreme case of a column. A wedge, described below, is a block specialization where the depth at one end is much smaller than that at the other end. This same specialization can be applied to a plate or column to produce a blade.

2.2.5 Wedge

A wedge is a block where the size at one end is zero, similar to the cone for cylinders. An idealized wedge is illustrated in Fig. 2.26.

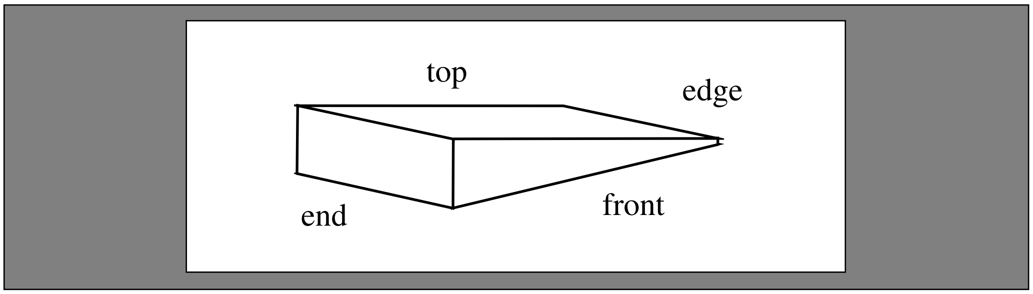


Figure 2.26 Idealized wedge.

Wedge Axes

The dimension axes for a wedge are identical to those for a block.

Wedge Center of Gravity

The wedge CG is located at the mid-point along the width and depth, and 1/3 from the end along the length.

Wedge Regions

The wedge has the same types but different numbers of regions than the block. There are five faces, six corners, and nine edges. Each face has a center. The locations of the wedge regions, with respect to the wedge CG, are shown in Table 2.6.

Name	Dimension	Value
top	along-depth	far-left
bottom	along-depth	far-right
front	along-width	far-left
back	along-width	far-right
edge	along-length	far-left
end	along-length	far-left

Table 2.6 Wedge Regions

Edges and corners are defined the same as for the block. The edge (edge) region is of primary interest, and is represented by the boundary between the top and bottom faces.

Wedge Size

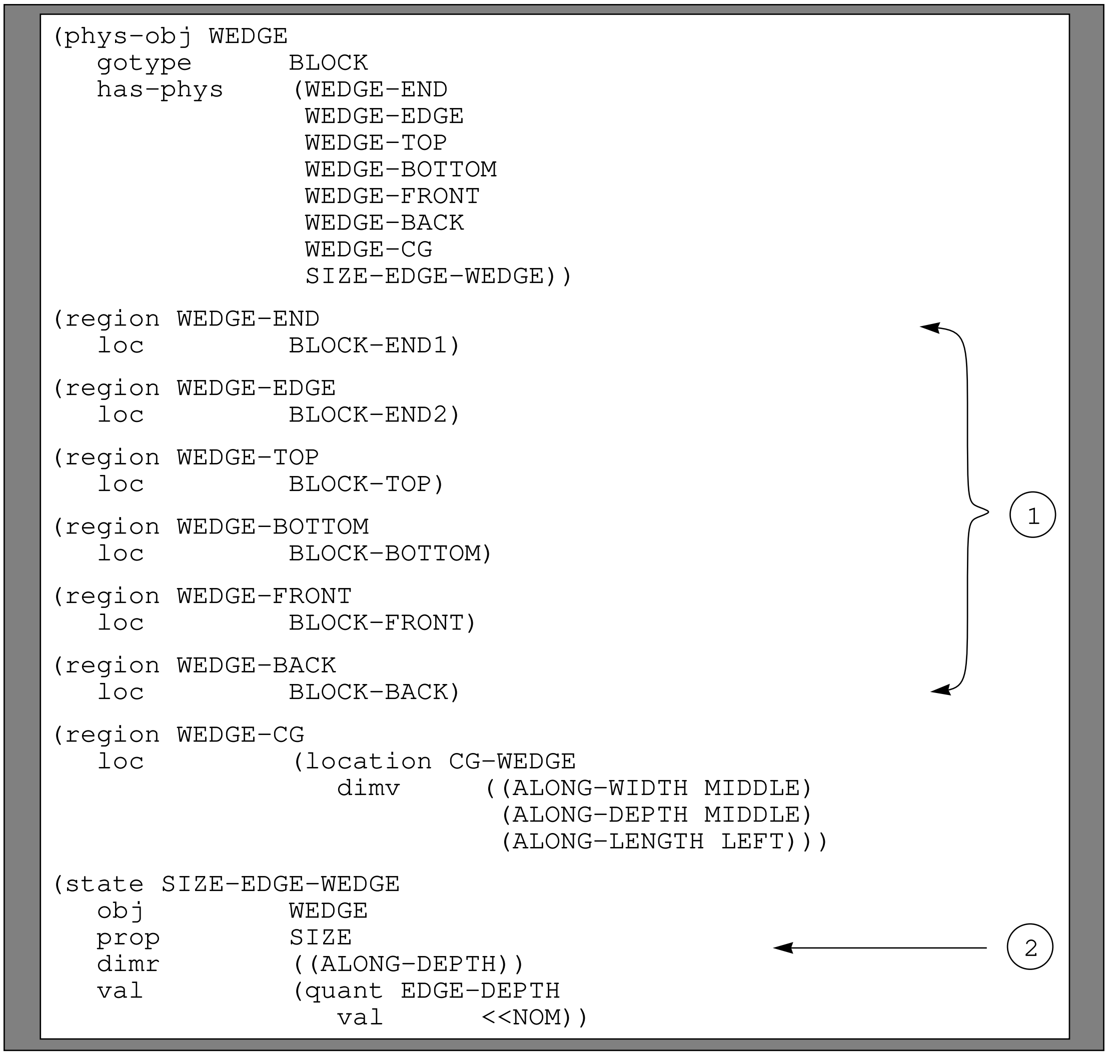
The wedge size is the same as the block, except that the depth on the edge end is zero.

Wedge Shape

The wedge is rectangular when viewed along its length and along its depth, and triangular when viewed along its width.

Wedge Representation

The wedge representation is illustrated with the regions instantiated (1) for the block The size is represented, with respect to the size in along-depth, as <<NOM.



```
(phys-obj WEDGE
   gotype      BLOCK
   has-phys    (WEDGE-END
                WEDGE-EDGE
                WEDGE-TOP
                WEDGE-BOTTOM
                WEDGE-FRONT
                WEDGE-BACK
                WEDGE-CG
                SIZE-EDGE-WEDGE))

(region WEDGE-END
   loc         BLOCK-END1)

(region WEDGE-EDGE
   loc         BLOCK-END2)

(region WEDGE-TOP
   loc         BLOCK-TOP)

(region WEDGE-BOTTOM
   loc         BLOCK-BOTTOM)

(region WEDGE-FRONT
   loc         BLOCK-FRONT)

(region WEDGE-BACK
   loc         BLOCK-BACK)

(region WEDGE-CG
   loc         (location CG-WEDGE
                  dimv     ((ALONG-WIDTH MIDDLE)
                            (ALONG-DEPTH MIDDLE)
                            (ALONG-LENGTH LEFT)))

(state SIZE-EDGE-WEDGE
   obj         WEDGE
   prop        SIZE
   dimr        ((ALONG-DEPTH))
   val         (quant EDGE-DEPTH
                  val      <<NOM))
```

Figure 2.27 Representation of wedge geometric primitive.

2.3 Object Primitives

In FONM, devices are described as compositions of object primitives rather than geometric primitives. Object primitives represent objects at a level where they are commonly recognized, so most are spatially more complex than a geometric primitive, and all are associated with a device function. An object primitive may be described as a composition of many geometric primitives, or as a single geometric primitive (or object primitive) and a number of specialized regions. Although objects presented in the following sections will be illustrated in both ways, the intent is to show the complexity of a geometric approach over a region approach. Finite element models use a geometric approach. FONM objects are all represented with a region approach.

2.3.1 Object Classification

There are eleven primitive objects, one for each for the objects which can instantiate a machine primitive. Idealized versions of these objects are illustrated in Fig. 2.28

2.3.2 Linkage-Object

A linkage-object is any object which can instantiate a linkage machine primitive. A linkage-object is any of the simple geometric objects described. For the purposes of this dissertation, a linkage object will be illustrated with a column (i.e., a thin rectangle, shown in Fig. 2.29).

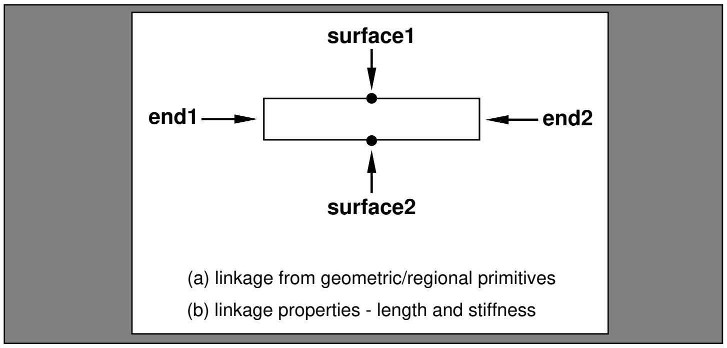
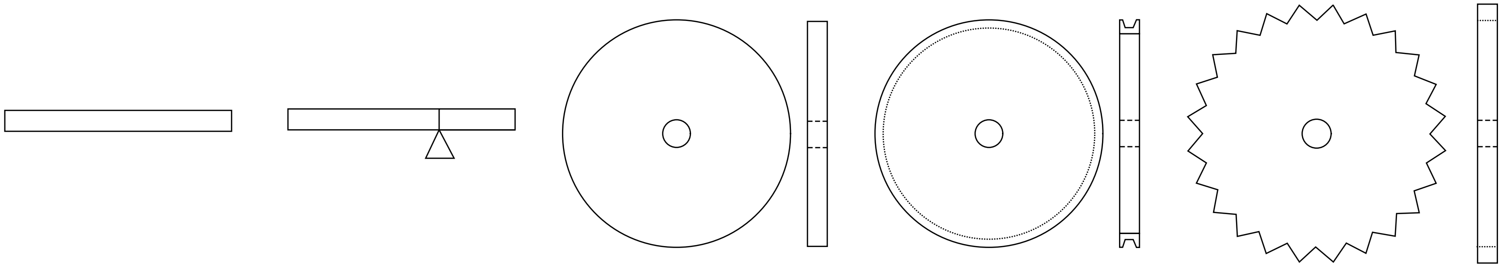


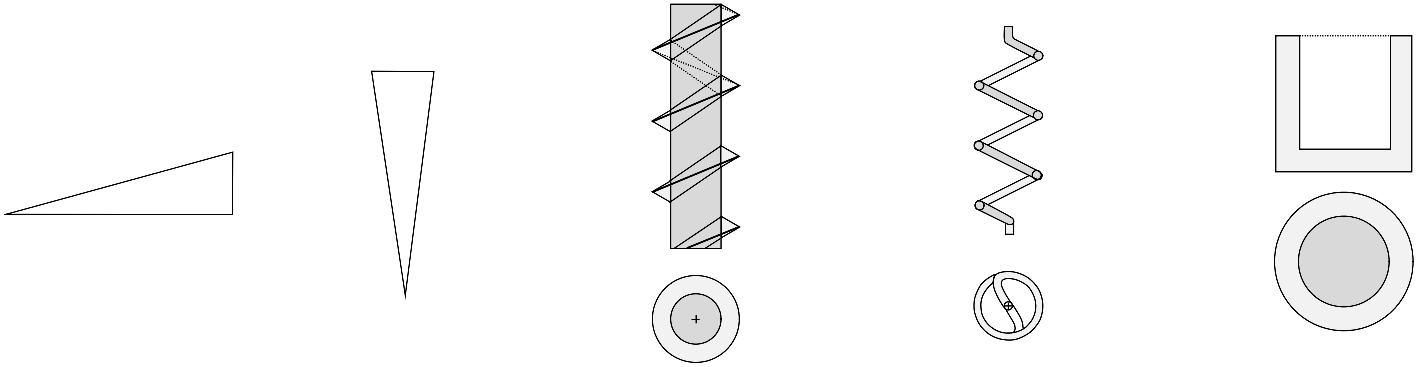
Figure 2.29 An idealized primitive linkage object. The linkage is illustrated with a rectangle (representing a column).

Linkage-objects are described with four regions, two ends and two surface points which represent the entire outer surface along the linkage length, regardless of its shape. These regions are identical to those of an idealized column.

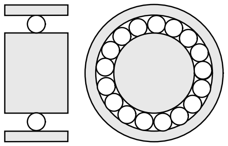
The linkage-object is represented simply as an object with a material stiffness property



(a) linkage (b) lever (c) wheel (d) pulley (e gear



(f) plane (wedge) (g) blade (wedge) (h) screw (i) spring (j) container



(k) bearing

Figure 2.28 Primitive objects in FONM.

value >NOM, where nominal is defined as a value which enables force transmission

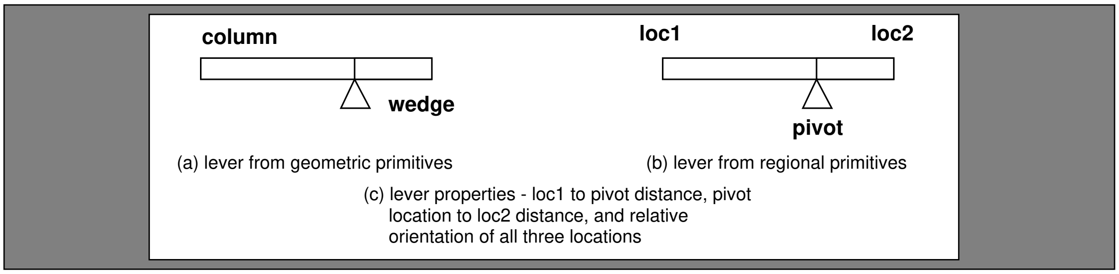
```
(phys-obj LINKAGE-OBJECT
   has-phys     (MS-LINKAGE-STIFFNESS
                 LINKAGE-END1
                 LINKAGE-END2))
(state MS-LINKAGE-STIFFNESS
   obj          LINKAGE-OBJECT
   prop         MATERIAL-STIFFNESS
   val          (quant ELASTIC-MS
                   units    PPI
                   val      >NOM))
(region LINKAGE-END1
   loc          (location LOC1-LINKAGE
                   dimv     ((?DIM      FAR-LEFT))))
(region LINKAGE-END2
   loc          (location LOC2-LINKAGE
                   dimv     ((?DIM      FAR-RIGHT))))
```

Figure 2.30 Representation of the linkage object primitive.

The linkage-object end locations are represented with respect to a variable dimension, since there is no geometric requirement that a linkage-object be a column, only that the ends be at extreme values in the dimension chosen.

2.3.3 Lever-Object

A lever-object is instantiated by any linkage object primitive with the addition of a pivot or fulcrum region. An idealized version, which will be used to illustrate devices in this dissertation, is shown in Fig. 2.31:



(a) lever from geometric primitives

(b) lever from regional primitives

(c) lever properties - loc1 to pivot distance, pivot location to loc2 distance, and relative orientation of all three locations

Figure 2.31 Idealized primitive lever objects, (a) based on composition of geometric primitives, and (b) based on a single geometric primitive and regions.

Two versions of the lever-object are shown in Fig. 2.31, one based on the composition of an idealized column and wedge geometric primitives, and one based on a linkage primitive object and three regions. In FONM, the latter description is used for representing devices, because it does not enforce the notion of a particular shape on how the primitive is instantiated.

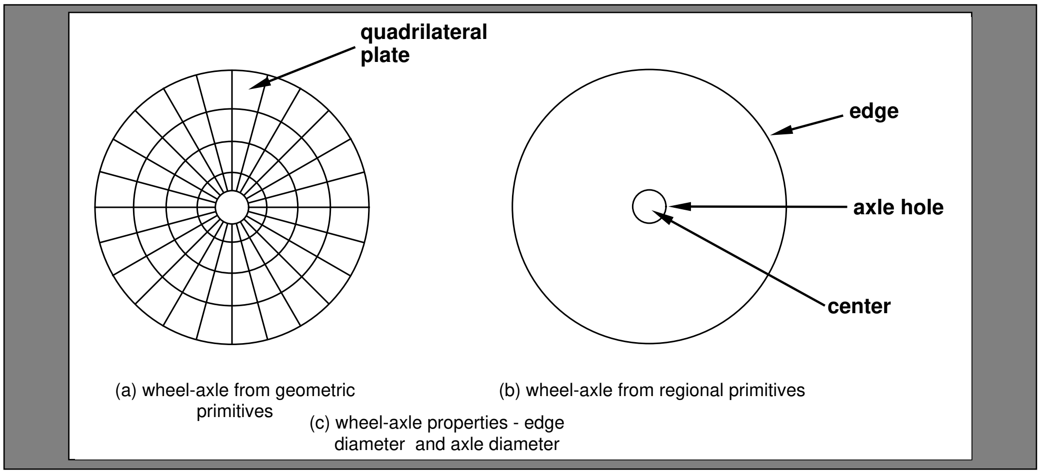
The lever-object regions are represented as instantiations of the linkage-object regions plus the addition of the pivot region which can be located anywhere along the dimension defined by the loc1 and loc2 regions (Fig. 2.32).

```
(phys-obj LEVER-OBJECT
   isa          LINKAGE
   has-phys     (LEVER-LOC1
                 LEVER-PIVOT
                 LEVER-LOC2))

(region LEVER-LOC1
   loc          LOC1-LINKAGE)

(region LEVER-PIVOT
   loc          (location LOC-LEVER-PIVOT
                   dimv      ((?dim       (LEFT NEAR-LEFT MIDDLE
                                           NEAR-RIGHT RIGHT)))))

(region LEVER-LOC2
   loc          LOC2-LINKAGE)
```

Figure 2.32 Representation of the lever object primitive.

2.3.4 Wheel-Object

A wheel-object is an object which can instantiate a wheel-axle machine primitive. The difference between a primitive object described with geometric objects and one described with a single geometric object and regions becomes more apparent with the introduction of the wheel. A wheel is an object which has a circular cross section, as depicted in Fig. 2.33.



(a) wheel-axle from geometric primitives

(b) wheel-axle from regional primitives

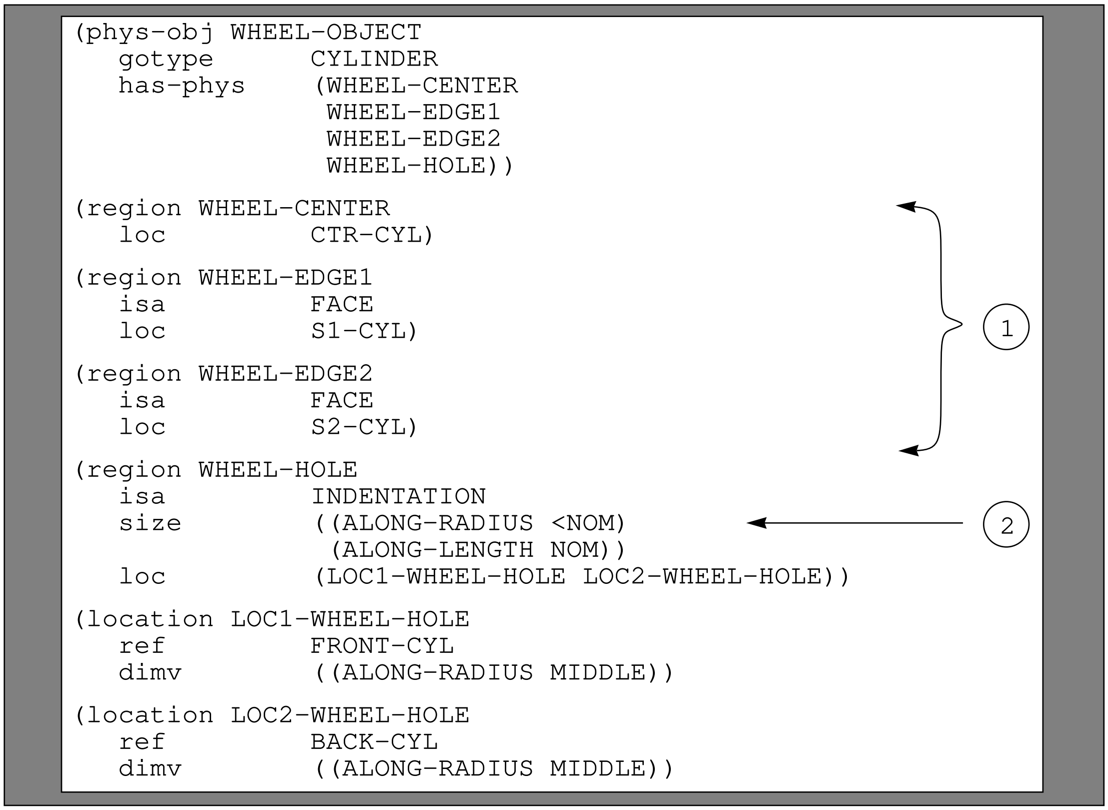
(c) wheel-axle properties - edge diameter and axle diameter

Figure 2.33 Idealized wheel primitive objects, (a) based on composition of geometric primitives, and (b) based on a single geometric primitive and regions.

The wheels shown in Fig. 2.33a and b differ in how many components are required to represent them, and in what types of inference they will support. The wheel in Fig. 2.33a is comprised of many quadrilateral plates connected together. This is identical to a simple finite element mesh. By representing the wheel in this fashion, the entire object is discretized and simulations can be performed on the position of every corner point of every plate. The wheel in Fig. 2.33b is described as a single geometric object, a cylinder or cylindrical plate, and four regions: two edge positions, a center, and an axle hole. These regions are identical to those of an idealized cylinder with the addition of the hole. This latter definition cannot be used to perform detailed simulations, because it represents the edge as a continuous surface, but it can be used to describe global behavior.

Just as important as the number of objects required to describe the wheel primitive is the ability of the wheel-object to inherit and share characteristics with the lever-object. Since the lever-object has two outer edge regions and a pivot region, these can be associated with the center and edge locations on the wheel-object. The same cannot be done between the versions based on geometric composition. The inheritance of these regions is represented below (1).

The wheel-object hole is also represented below. The size along-radius is represented as <NOM, meaning that the hole must be smaller than the size of the wheel. The size along-depth is represented as NOM, since the hole penetrates the entire wheel.

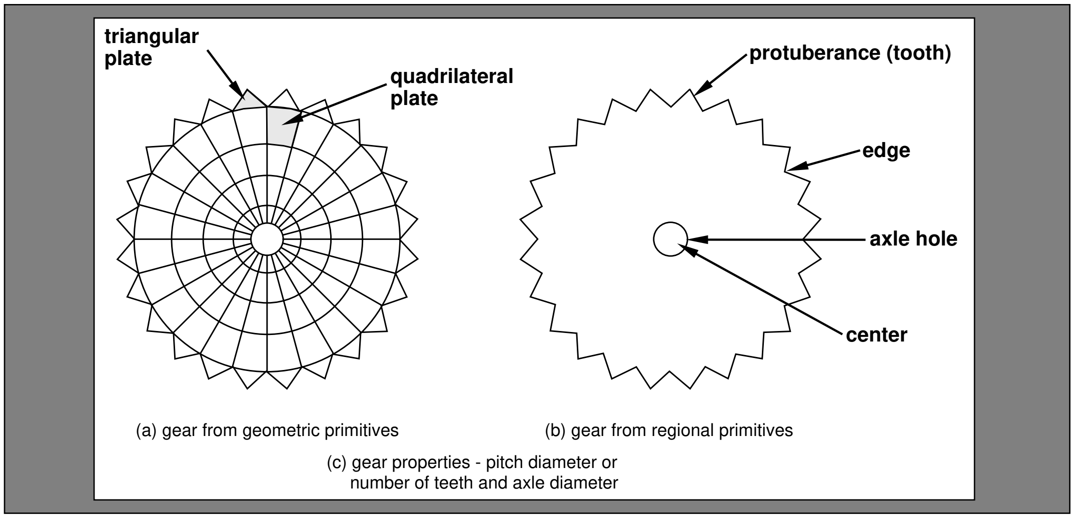


```
(phys-obj WHEEL-OBJECT
   gotype      CYLINDER
   has-phys    (WHEEL-CENTER
                WHEEL-EDGE1
                WHEEL-EDGE2
                WHEEL-HOLE))

(region WHEEL-CENTER
   loc         CTR-CYL)

(region WHEEL-EDGE1
   isa         FACE
   loc         S1-CYL)

(region WHEEL-EDGE2
   isa         FACE
   loc         S2-CYL)

(region WHEEL-HOLE
   isa         INDENTATION
   size        ((ALONG-RADIUS <NOM)
                (ALONG-LENGTH NOM))
   loc         (LOC1-WHEEL-HOLE LOC2-WHEEL-HOLE))

(location LOC1-WHEEL-HOLE
   ref         FRONT-CYL
   dimv        ((ALONG-RADIUS MIDDLE))

(location LOC2-WHEEL-HOLE
   ref         BACK-CYL
   dimv        ((ALONG-RADIUS MIDDLE))
```

Figure 2.34 Representation of the wheel object primitive.

2.3.5 Gear-Object

A gear-object is an object which can instantiate a gear machine primitive. A gear-object is comprised of two gears, or a gear and a rack, in tooth contact. The gear primitive object is modified from the wheel-object by adding teeth to the outer edge, as shown with the idealized versions in Fig. 2.35



(a) gear from geometric primitives

(b) gear from regional primitives

(c) gear properties - pitch diameter or number of teeth and axle diameter

Figure 2.35 Idealized gear primitive objects, (a) based on composition of geometric primitives, and (b) based on a single geometric primitive and regions.

In Fig. 2.35a, the gear tooth is represented with the addition of many triangular plates along the outer edge. In Fig. 2.35b, the wheel-object edge region is specialized as a protuberance. The figure is a bit erroneous, since the gear-object tooth regions are really represented with two regions as with the wheel-object. The figure is left as is to illustrate that the tooth protuberances can be anywhere along the edge without affecting the representation. The gear properties described in Fig. 2.35c: the gear-object's pitch diameter or number of teeth, and the axle diameter, are the only property values needed to make predictions about a gear's behavior. The pitch diameter can be obtained from the two tooth regions without the need to have explicit information about the number of teeth on the gear-object.

The representation for the gear-object is shown below, and represents the tooth regions

```
(phys-obj GEAR-OBJECT
   isa          WHEEL-OBJECT
   has-phys     (GEAR-CENTER
                 GEAR-TOOTH1
                 GEAR-TOOTH2
                 GEAR-HOLE))

(region GEAR-CENTER
   loc          WHEEL-CENTER)

(region GEAR-TOOTH1
   isa          PROTUBERANCE
   loc          WHEEL-EDGE1)

(region GEAR-TOOTH2
   isa          PROTUBERANCE
   loc          WHEEL-EDGE2)

(region GEAR-HOLE
   isa          INDENTATION
   loc          WHEEL-HOLE))
```

Figure 2.36 Representation of the gear object primitive.

as protuberances located at wheel-object edge regions.

2.3.6 Pulley-Object

A pulley-object is an object which can instantiate a pulley machine primitive. A pulley- object is comprised of two components: a pulley and a belt or rope. An idealized pulley is illustrated in Fig. 2.37 below.

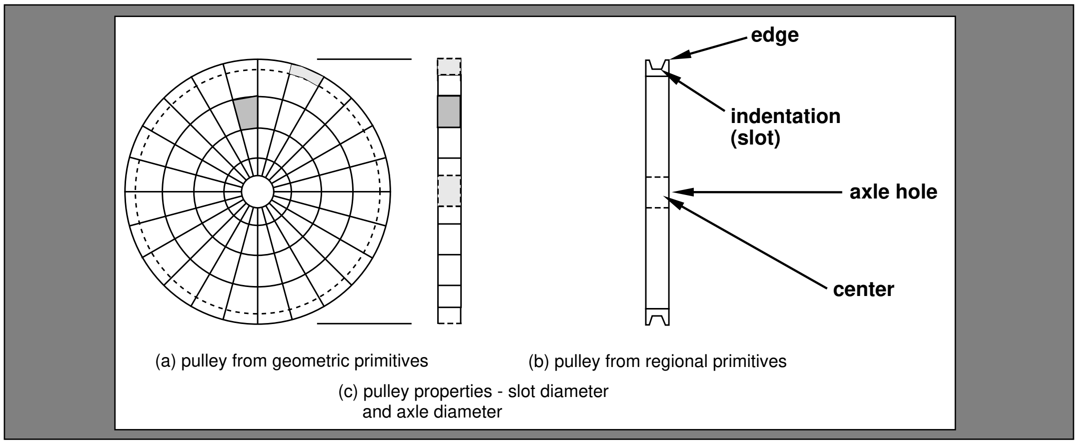


Figure 2.37 Idealized pulley primitive objects, (a) based on composition of geometric primitives, and (b) based on single geometric primitive and regions.

As with the gear-object, Fig. 2.37a, b, and c represent a combined geometric object version, a region based version, and the properties needed for behavioral analysis, respectively. The version in Fig. 2.37a replaces the slot and axle of a real pulley with solid plates that have proportional mass properties. The version in Fig. 2.37b represents the edge and hole as indentation regions. Indentations are associated with behavioral containment, so these regions capture the function of the pulley.

The rope or belt is represented as a one-direction (along-length, away from CG) linkage. The belt can transmit mechanical force in both directions along its length.

```
(phys-obj PULLEY-OBJECT
   isa         WHEEL-OBJECT
   has-phys    (PULLEY-CENTER
                PULLEY-SLOT1
                PULLEY-SLOT2
                PULLEY-HOLE))

(region PULLEY-CENTER
   loc         WHEEL-CENTER)

(region PULLEY-SLOT1
   isa         INDENTATION
   loc         WHEEL-EDGE1)

(region PULLEY-SLOT2
   isa         INDENTATION
   loc         WHEEL-EDGE2)

(region PULLEY-HOLE
   isa         INDENTATION
   loc         WHEEL-HOLE)
```

Figure 2.38 Representation of the pulley object primitive.

The pulley-object representation in Fig. 2.38 shows the slot regions being represented as indentations at the wheel edge locations, analogous to the gear-object.

2.3.7 Bearing-Object

A bearing-object is an object which can instantiate a bearing machine primitive. A bearing-object can be instantiated by an object with a circular cross section. An idealized bearing-object is illustrated in Fig. 2.39.

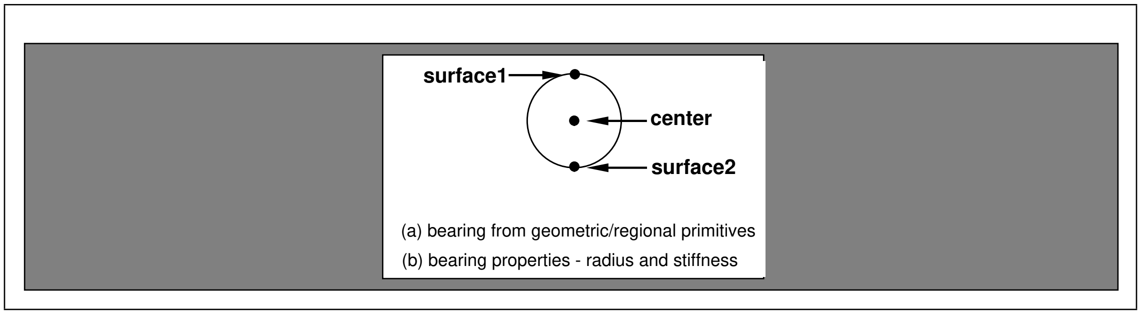


Figure 2.39 Idealized bearing primitive object. A bearing is a cylinder, a cone, or a sphere geometric primitive.

Like the wheel-object, the object in Fig. 2.39 is represented with the cylinder and sphere regions. Unlike the wheel-object, the bearing-object does not have an axle hole region. There is no access to the center of the object, so force and motion are transmitted on the surface alone.

```
(phys-obj BEARING-OBJECT
   gotype      (SPHERE CYLINDER CONE)
   isa         WHEEL-OBJECT
   has-phys    (BEARING-CENTER
                BEARING-S1
                BEARING-S2))

(region BEARING-CENTER
   loc         WHEEL-CENTER)

(region BEARING-S1
   loc         WHEEL-EDGE1)

(region BEARING-S2
   loc         WHEEL-EDGE2)
```

Figure 2.40 Representation of the bearing object primitive.

The representation for the bearing object allows for a conical shape as well as a spherical or cylindrical one, since the bearing functions by transmitting force passively through its surface regions rather than actively as with the other wheel-like objects.

2.3.8 Plane-Object

A plane-object is an object which can instantiate the plane machine primitive. An idealized version of the plane-object, for use in illustrating device representations in FONM, is shown in Fig. 2.41.

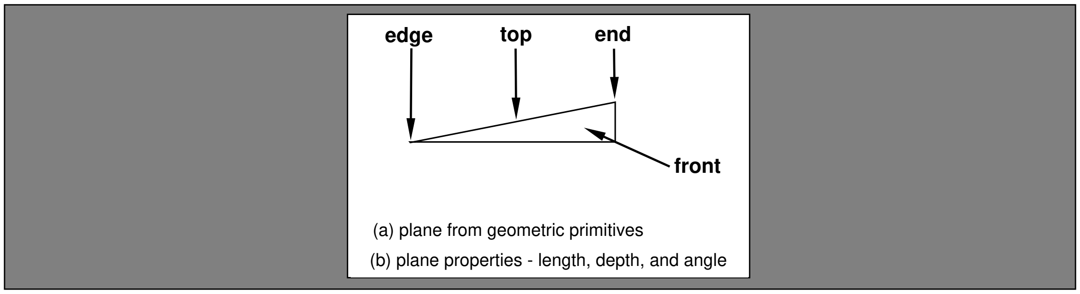


Figure 2.41 Idealized plane primitive object. A plane is a wedge geometric primitive.

The idealized plane-object is represented with a wedge geometric primitive. In general, a plane-object can be instantiated by an object that is offset with respect to the path of motion, so the triangular shape is an implied, rather than an explicit geometric, requirement of the object. The plane-object is represented with two face regions and an edge region. The plane of motion is the top (face) region, and the end (face) and edge (edge) regions determine the offset.

```
(phys-obj PLANE-OBJECT
   gotype      WEDGE
   has-phys    (PLANE-END
                PLANE-EDGE
                PLANE-TOP))

(region PLANE-END
   loc         WEDGE-END)

(region PLANE-EDGE
   loc         WEDGE-EDGE)

(region PLANE-TOP
   loc         WEDGE-TOP)
```

Figure 2.42 Representation of the plane object primitive.

2.3.9 Blade Object

A blade-object is an object which can instantiate the blade machine primitive. A blade- object, unlike a plane-object, has an explicit triangular shape requirement, as illustrated by the idealized blade in Fig. 2.43.

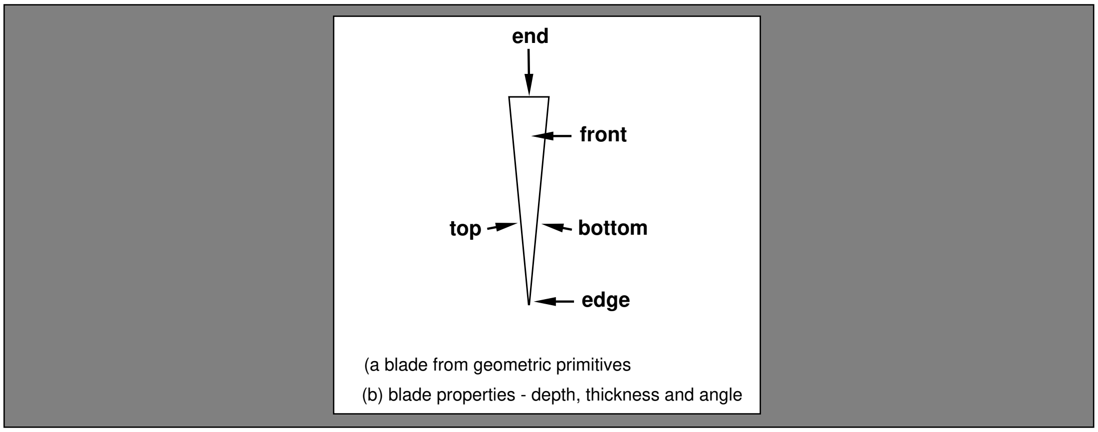


Figure 2.43 Idealized blade primitive object. The blade is a wedge geometric primitive.

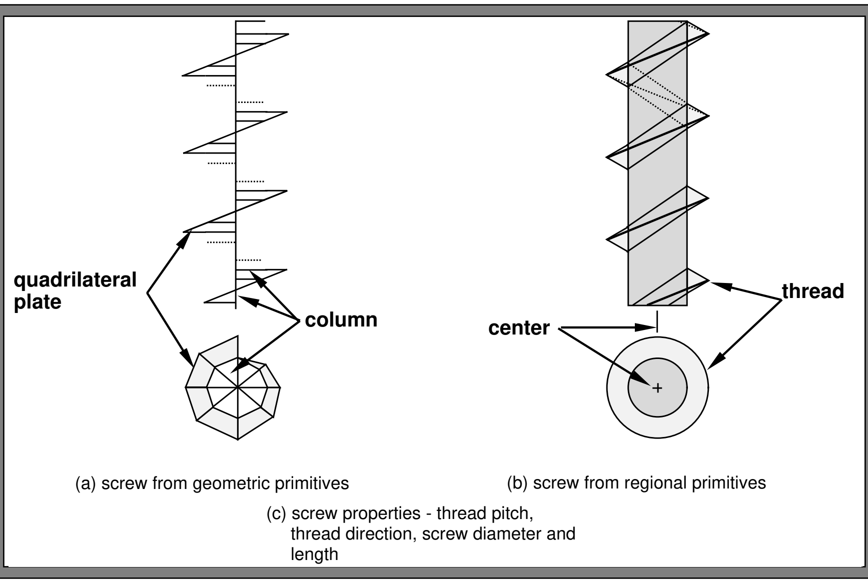
The blade-object is represented with a wedge geometric primitive, so their regions are identical.

```
(phys-obj BLADE-OBJECT
   gotype      WEDGE
   has-phys     (BLADE-END
                 BLADE-EDGE
                 BLADE-TOP
                 BLADE-BOTTOM
                 BLADE-FRONT
                 BLADE-BACK))

(region BLADE-END
   loc         WEDGE-END)

(region BLADE-EDGE
   loc         WEDGE-EDGE)

(region BLADE-TOP
   loc         WEDGE-TOP)

(region BLADE-BOTTOM
   loc         WEDGE-BOTTOM)

(region BLADE-FRONT
   loc         WEDGE-FRONT)

(region BLADE-BACK
   loc         WEDGE-BACK)
```

Figure 2.44 Representation of the blade object primitive.

2.3.10 Screw-Object

A screw-object is an object which can instantiate a screw machine primitive. The screw object can be visualized as a cylinder or cone geometric primitive with a triangular column wrapped around its edge, as depicted in Fig. 2.45.



(a) screw from geometric primitives

(b) screw from regional primitives

(c) screw properties - thread pitch, thread direction, screw diameter and length

Figure 2.45 Idealized screw primitive object, (a) based on composition of geometric primitives, and (b) based on single geometric primitive and regions.

Two versions are shown. Fig. 2.45a represents a screw as a helical series of columns cantilevered off a central column, with quadrilateral plates connected to the columns. This is a finite element representation of a screw. Fig. 2.45b represents the screw as a cylinder whose edge is a protuberance, like a gear, formed from a wedge wrapped around it, as shown in Fig. 2.46a. Another way of thinking about the screw-object representation is as a cylinder with a wedge-shaped column wrapped around it as shown in Fig. 2.46b. The first figure illustrates why a helical shape need not be described, while the latter figure illustrates how a thread can be represented so as to account for cutting without changing the geometric primitives used to describe the object

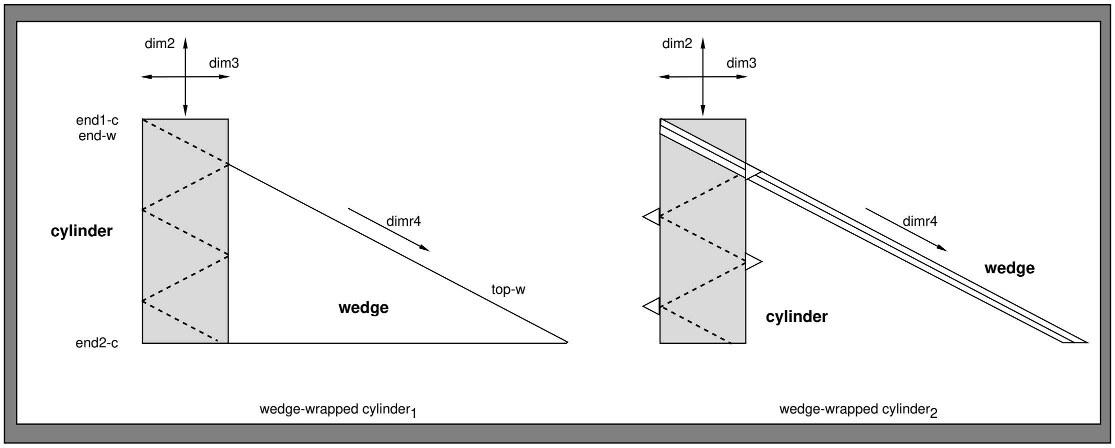
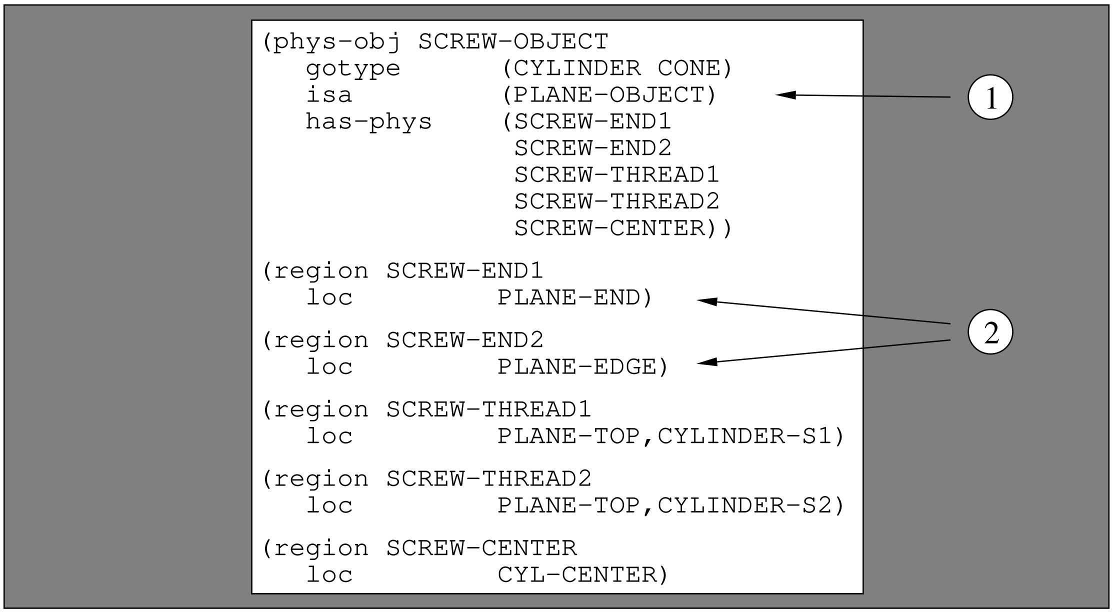


Figure 2.46 Concept of a screw as a combination of a cylinder and a wedge.

The screw-object is represented with a cylinder (or cone) which inherits properties from a plane-object (at 1 in Fig. 2.47). The screw-object regions are the two cylinder ends, a center, and two surface regions.



```
(phys-obj SCREW-OBJECT
   gotype       (CYLINDER CONE)
   isa          (PLANE-OBJECT)
   has-phys     (SCREW-END1
                 SCREW-END2
                 SCREW-THREAD1
                 SCREW-THREAD2
                 SCREW-CENTER))

(region SCREW-END1
   loc          PLANE-END)

(region SCREW-END2
   loc          PLANE-EDGE)

(region SCREW-THREAD1
   loc          PLANE-TOP,CYLINDER-S1)

(region SCREW-THREAD2
   loc          PLANE-TOP,CYLINDER-S2)

(region SCREW-CENTER
   loc          CYL-CENTER)
```

Figure 2.47 Slot-filler representation of the screw object.

The ends are related to the end and edge of the plane-object (2), respectively. The threads are related to the plane-object top region and the cylinder s1 and s2 regions.

2.3.11 Spring-Object

A spring-object is an object which can instantiate a spring machine primitive. A spring- object can be instantiated by any elastic object, so any linkage-object can instantiate a spring-object. An idealized spring-object is depicted as a coiled filament geometric object in Fig. 2.48.

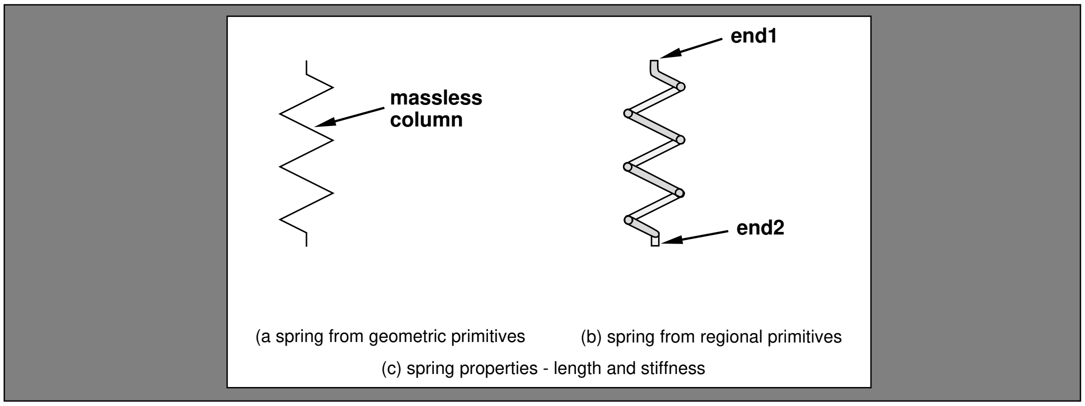


Figure 2.48 Idealized spring primitive objects, (a) based on composition of geometric primitives, and (b) based on single geometric primitive and regions.

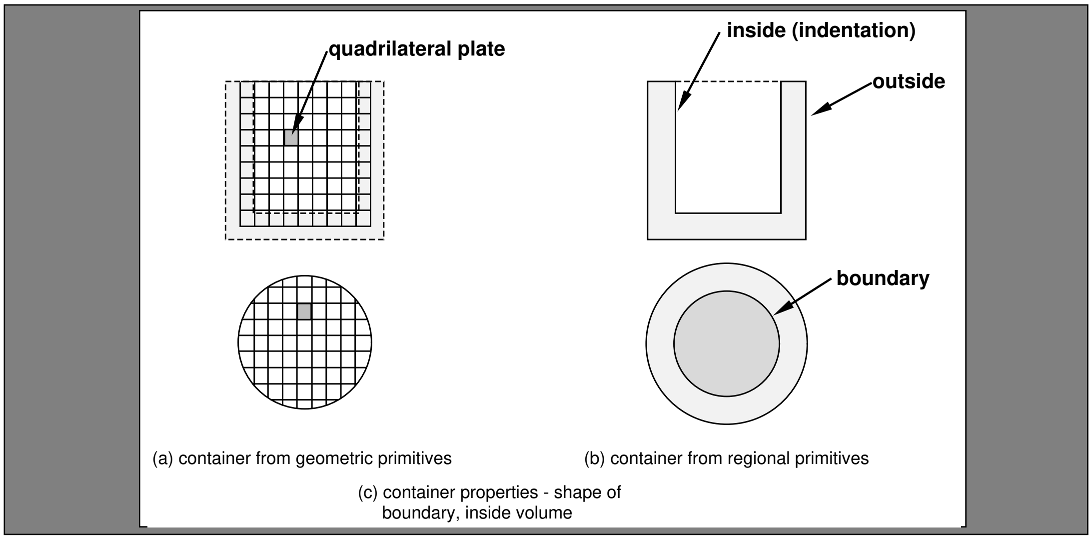
An idealized spring, based on a finite element version, is shown in Fig. 2.48a. In this version, the spring is represented with a massless column which has a material stiffness. The version in Fig. 2.48b is represented as a filament wrapped around a cylindrical shape, with the two cylinder end regions.

```
(phys-obj SPRING-OBJECT
   isa         LINKAGE-OBJECT
   has-phys    (SPRING-END1
                SPRING-END2))

(region SPRING-END1
   loc         LINKAGE-END1)

(region SPRING-END2
   loc         LINKAGE-END2)
```

Figure 2.49 Representation of the spring object primitive.

2.3.12 Container-Object

A container-object is an object which can instantiate a container machine primitive. A container-object is instantiated by an object with an indentation, as depicted by the idealized version in Fig. 2.50.



(a) container from geometric primitives

(b) container from regional primitives

(c) container properties - shape of boundary, inside volume

Figure 2.50 Idealized primitive container objects, (a) based on composition of geometric primitives, and (b) based on single geometric primitive and regions.

Two container-objects are depicted in Fig. 2.50a and b. Fig. 2.50a is a container comprised of a triangular and quadrilateral plates connected together in a tube. This tube is connected to another set of triangular and quadrilateral plates which are themselves connected together to form a circular plate. The two generalized objects form a vessel. Fig. 2.50b represents the container-object as a cylinder geometric object with an indentation region along its length and coaxial with its center region. The surface regions are called the container-object *outside*, and the indentation forms the container-object *inside*. The boundary between the two is called the container-object *boundary*

```
(phys-obj CONTAINER-OBJECT
   isa         LINKAGE-OBJECT
   has-phys    (CONTAINER-INSIDE
                CONTAINER-OUTSIDE
                CONTAINER-BOUNDARY))

(region CONTAINER-INSIDE
   isa         INDENTATION)

(region CONTAINER-OUTSIDE
   loc         LINKAGE-END1)

(region CONTAINER-BOUNDARY
   loc         LINKAGE-END2)
```

Figure 2.51 Representation of the container object primitive.

2.4 Devices

In FONM, a device's components are each represented with one or more object primitives. A device may also have additional region specifiers at the region locations of its object primitive components. Consider the nutcracker depicted in Fig. 2.52. The nutcracker is a device comprised of four components. The handles (LEVER1, LEVER2) are represented with two lever-objects, the pivot piece (PIVOT) is represented with a linkage-object, and the spring (SPRING) is represented with a spring-object.

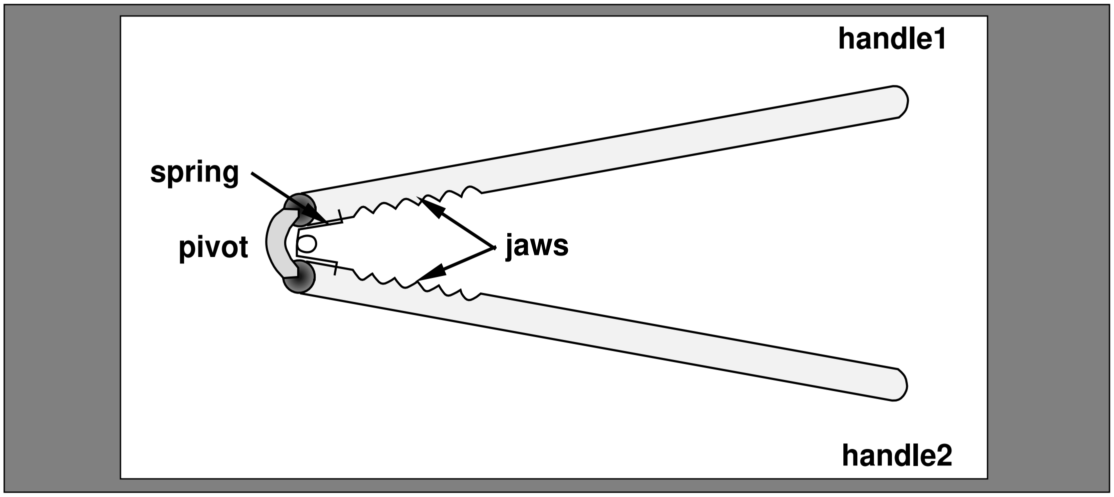


Figure 2.52 Four components in a nutcracker: two levers, a pivot, and a spring.

Each nutcracker component has region specifiers which are associated with, and directly support the component's role in a particular nutcracker function. For example, when used for cracking nuts, the nutcracker is a hand-operated device, so it must have some location (e.g., HANDLE-L1 on LEVER1) to operate the device (i.e., by grasping and pressing). It must also have some location where the force which cracks the nut is produced. Force can be transmitted at contact locations, so the jaw region, JAW-L1 on LEVER1, wherever it is physically located, is a necessary region for nutcracking. LEVER1 is connected to PIVOT, so the location of that connection must also be represented. This location is called END-L1. These three regions instantiate the LOC1, LOC2, and PIVOT lever object regions. By specializing the shape (JAW-L1 is a protuberance) and size (HANDLE-L1 will fit the hand), the lever object regions take on the functions of those region types. The same set of relationships holds for LEVER2 (with HANDLE-L2, JAW-L2, and END-L2), PIVOT (END1-P and END2-P), and SPRING (END1-S and END2-S). The overall nutcracker device is represented with four components and ten regions.

In order to fully describe nutcracker statics, the spatial relationships between its components, and their connectivity, must also be represented.

2.5 Relational Characteristics

A relational characteristic describes a spatial relationship between two objects. Object relational characteristics constrain the interactions in which the objects can participate. For example, if the nutcracker handles illustrated in Fig. 2.52 weren't coplanar, it would be difficult to operate, let alone crack nuts with the device. *Coplanar* is an object orientation. Similarly, in some situations it is important to know how regions of different objects are positioned in space. This is called object *placement*. In FONM, object orientation and placement are the two relational characteristics used to represent spatial interaction.

2.5.1 Orientation

Orientation describes dimensional alignment between two objects and determines how they may interact. For example, a nut and bolt must have their centers aligned in order to thread them together. Objects whose rotational centers are aligned are *coaxial*. Orientation between objects is represented with a knowledge structure called *orient*. Orient has three roles: obj1, obj2, and dim. The *obj1* and *obj2* roles describe the objects concerned. The *dim* role describes the dimensions shared by obj1 and obj2, or the offset angle between them. When the dim slot has multiple elements, then each element represents a shared dimension. For example, a nut and bolt, when connected, are both colinear and coaxial, because they share longitudinal and rotational axes.

```
(orient COAXIAL-N-B
  obj1     NUT
  obj2     BOLT
  dim      ((ALONG-LENGTH) (ABOUT-LENGTH))
```

Figure 2.53 Coaxial object orientation in FONM.

In this example, the dim role has two elements (ALONG-LENGTH) and (ABOUT-LENGTH), meaning that the nut longitudinal axis is the same as the bolt longitudinal axis, and the same for the rotational axis. The orientations which are represented in FONM are shown in Table 2.7.

Orientation Type	Dimensional Comparison
colinear	shared translational (length) axis
coaxial	shared rotational (center) axis
coplanar	two shared translational axes
coincident	all translational axes shared
parallel	length axis oriented same as global
perpendicular	axis offset is 90 degrees
skewed	axis offset is not 90 degrees

Table 2.7 Object orientations as a function of shared dimensions

Table 2.7 shows orientation between objects as a function of the number of shared dimensional axes. A ramp for loading goods onto a truck is leaned against the ground and against the rear of the truck. The orientation, called an *offset*, is represented by the dimension mismatch and the angle between them. For example, the ramp ALONG-LENGTH dimension makes an angle with respect to the ground.

```
(orient SKEW-LRAMP
  obj1     LOADING-RAMP
  obj2     GROUND
  dim      ((ALONG-LENGTH
             ALONG-LENGTH
             (quant SLRG
                units  degrees
                val    15)))
```

Figure 2.54 Skew object orientation in FONM.

In this example, the dim role has a single element, so there is only one dimension needed to describe the orientation between LOADING-RAMP and GROUND. The dimensions described are the same and the offset between them is represented by the quantity SLRG (Skew-LRamp-Ground, 15 degrees)

2.5.2 Placement

To represent placement, a frame of reference is required. For example, a person would say that a book which is supported by a table is above the table, because we assume earth as a frame of reference. In FONM, the frame of reference must be made explicit, so in the example above, the dimension of comparison is earth radial and the table is closer to earth's center than the book. The term *above* describes the relative placement of the book and table with respect to a reference dimension and location. Placement and location are different. Location is used to locate regions of an object, whereas placement locates unique objects, either with respect to one another or to a global reference frame. Location can be used when two regions of unique objects are in contact, since their positions will be coincident and their spatial orientation won't matter as long as they are in contact. When objects are not in contact, placement must be used, because the orientation of one object with respect to an inertial frame of reference may change while the orientation of the other object may not.

Placement is represented with a knowledge structure called *place*, which has four roles: obj, odimr, ref, and rdimv. The *obj* role describes the region on the object (e.g., a book) being located. The *odimr* role describes the dimension and direction of the object region with respect to the reference object. The *ref* role describes the reference object or region (e.g., a table) from which the placement of the named object will be determined. The *rdimv* role describes the pertinent dimension and qualitative value with respect to the reference object's static description. For example, the placement of the book *above* the table is represented below:

```
(place ABOVE-B-T
  obj      COVER-BOOK
  odimr    (ALONG-DEPTH NEG)
  ref      TOP-TABLE
  rdimv    (ALONG-DEPTH FAR-LEFT)
```

Figure 2.55 Representation of the 'above' object placement in FONM.

In Fig. 2.55, the object being placed is BOOK, and the region COVER-BOOK is the location, on the book, which is being located in the reference frame. The location of COVER-BOOK on BOOK is represented with the same dimension as the placement comparison, ALONG-DEPTH, however, that is not generally the case. The reference object is TABLE, and the region is TOP-TABLE. The reference location is described as ((ALONG-DEPTH FAR-LEFT)). The meaning of ABOVE-B-T is that the book cover's relationship to the middle of the table top is along the book's depth dimension and toward the table top. The middle is assumed because no other dimensions or values are represented in the placement, so the CG locations in those dimensions are used.

In FONM, placement is primarily a function of dimension, in earth coordinates, as shown in Table 2.8

Placement Type	Reference DImension	Reference Value
ABOVE	vertical (along-radius)	left, near-left, middle, near-right, right, far-right
BELOW	vertical (along-radius)	far-left, left, near-left, middle, near-right, right
LEFT-OF	horizontal (about-radius)	far-left, left, near-left, middle, near-right, right
RIGHT-OF	horizontal (about-radius)	left, near-left, middle, near-right, right, far-right

Table 2.8 Placement in FONM.

In Table 2.8, the reference value defaults to ground level (FAR-LEFT), so an object sitting on the ground is above the ground if the reference value is anything but (FAR-LEFT). When the reference object is not earth, then the object dimension which coincides with earth vertical is used for representing above and below for other objects.

2.6 Object Connectivity

A mechanical device is comprised of components connected together in spatially meaningful ways. In FONM, object connectivity is represented as a mutual restraint state on two objects, where the local region and restraint dimension are identical. Object connectivity is understood by the degrees of freedom which are restrained by the presence of each object. For example, consider two wood blocks which are bolted together along their depth, as shown in Fig. 2.56.

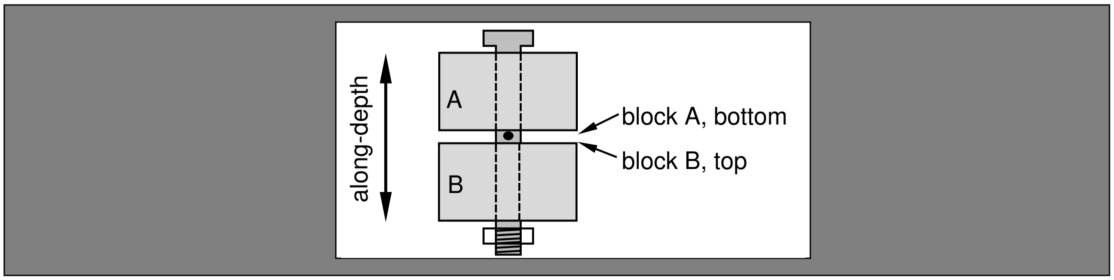


Figure 2.56 Object connection in FONM. Two bolted blocks.

Block A's bottom face is in contact with block B's top face. Both objects are restrained, by the other, or by the bolt (it doesn't matter for this example), in all dimensions. This is called a *clamped*, or *fixed*, connection. In FONM, the representation of connection states is directly a function of the number and type of restrained degrees of freedom, as shown in

Table 2.9.

Connection Name	Degrees of Freedom Restrained
free	None
contact	One direction
pivoted	One local translational direction
fitted	Two translational and one rotational
roller	Two translational and two rotational
pinned	All translational and two rotational
rigid	All local D.O.F.
clamped or fixed	All D.O.F.

Table 2.9 Connections in FONM.

2.7 Device Static Descriptions

A static device description, or representation, is comprised of the primary object for each component, its physical characteristics, and its relational characteristics to the other components. Device static representations fall into three categories: (1) simple devices, (2) compound devices, and (3) complex devices. A *simple* device is one where all the components are rigidly connected together and move as a single unit. A *compound* device is one where at least two components can move relative to one another. A *simple* compound device has two components which can move relative to one another, and a *multiple* compound device is one with many moving components, but each is a simple device. A *complex* device is comprised of systems, each of which can be a multiple compound device. In the following sections, static representations for devices in each of these categories will be presented. For each device, an illustration of what a real device of this type looks like is presented, along with an illustration of how FONM primitive objects are used to represent the device. A *static device diagram* (SDD) for each device is also presented, rather than fleshing out the full representation, and representations are presented where necessary to make a point. Each SDD presents the components and their connectivity, as detailed in the previous sections. Devices which are comprised of similar components or are connected in similar ways should have similar static representations regardless of how they behave and function. Device representations presented in this section will illustrate these similarities, and, in other chapters, the differences will be shown in dynamic representations of the same devices.

2.7.1 Simple Devices

A simple device has no moving components. Simple devices may have complex shapes, so their regional descriptions play an important role in how they behave and how they are viewed by problem solvers. Two simple devices are represented and described in this section: (1) a bottle opener, and (2) a screwdriver. Each device is comprised of regions which take on the characteristics of one or more primitive object. Four additional simple devices: (3) carving knife, (4) hammer, (5) shovel, and (6) spoon are represented in Appendix A.1.

Bottle Opener Statics

A bottle opener is a simple object which has a blunt end and a pointed end. At each end there is a piece which protrudes and enables the generation of leverage. An illustration of the true bottle opener shape is shown in Fig. 2.57.

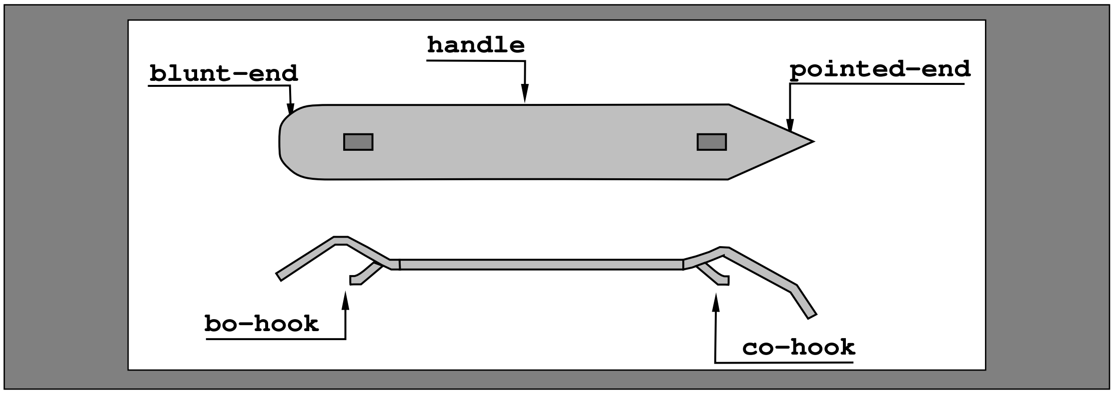


Figure 2.57 Bottle opener illustration.

Notice in the figure that the actual object is made of a single piece and that the shape at both ends is complex. The FONM idealization of the bottle opener is shown in Fig. 2.58 below.

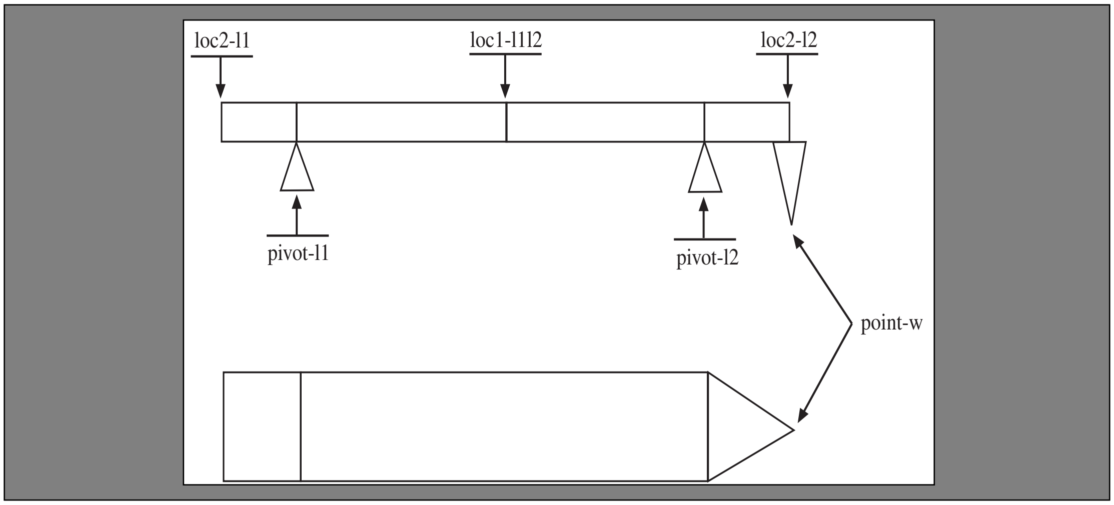


Figure 2.58 Bottle opener object primitives. Two lever-objects and a blade-object.

In Fig. 2.58, the device is shown as two lever-objects connected at their ends (`loc1-l1l2`). The blunt end of the bottle opener is represented with `loc2-l1`, its associated protrusion with `pivot-l1` and its body with `loc1-l1`. A similar relationship holds for the pointed end, accept a blade object is connected at `loc2-l2`. Three FONM objects are used to represent the device, two lever-objects and a blade-object, as shown in the static device diagram below (Fig. 2.59).

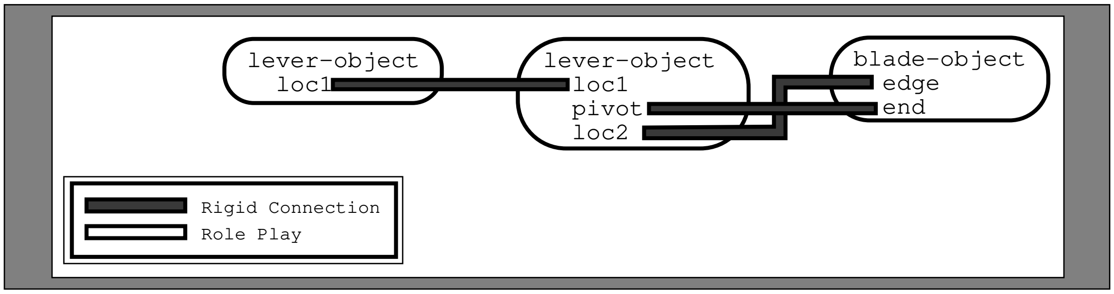
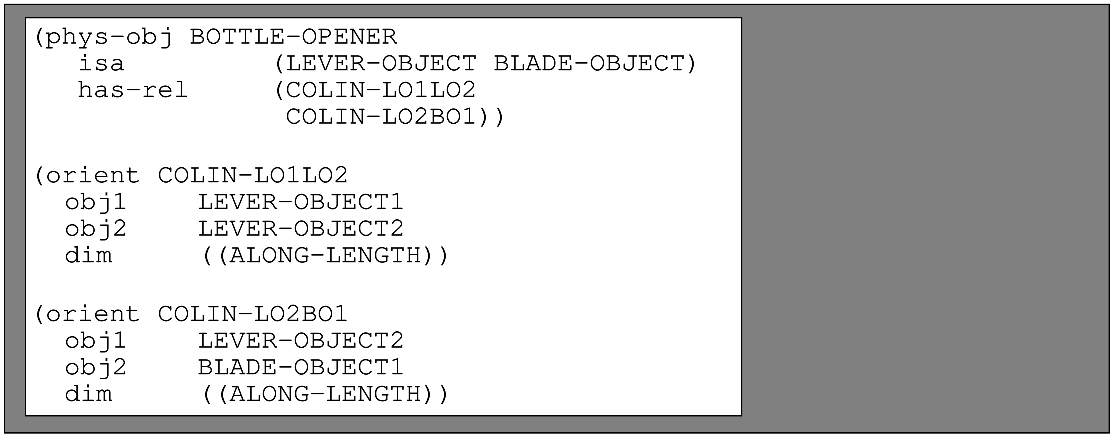


Figure 2.59 Bottle opener static device diagram (SDD).

A static device diagram illustrates the FONM primitive objects which comprise a device's components, and their connectivity as arcs. Both lever-objects share the same region, `loc-l1`. This is represented as a rigid connection between the two components at `loc-l1`, where the rigid connection is depicted as an arc with a diagonal stripe. The `pivot` region of the right-hand lever-object is also the `end` region of the blade-object. These regions are both shared between the two objects and the locations of additional rigid connections between the two objects.

The `pivot` and `loc2` regions for the left-hand lever-object are not shown here, since they are not important in illustrating the component connectivity. In addition, the fact that the protrusions of the bottle opener are represented with protrusions is not illustrated. The colinear spatial relations between these objects are represented below.



```
(phys-obj BOTTLE-OPENER
   isa         (LEVER-OBJECT BLADE-OBJECT)
   has-rel     (COLIN-LO1LO2
                COLIN-LO2BO1))

(orient COLIN-LO1LO2
  obj1    LEVER-OBJECT1
  obj2    LEVER-OBJECT2
  dim     ((ALONG-LENGTH))

(orient COLIN-LO2BO1
  obj1    LEVER-OBJECT2
  obj2    BLADE-OBJECT1
  dim     ((ALONG-LENGTH))
```

Figure 2.60 Spatial relationships for the bottle opener static representation.

Note that the bottle-opener is labeled as a lever-object and a blade-object. In FONM, when a simple device is represented with an object primitive, it inherits the static capabilities of that object. The most specific physically unique (geometric) object type will be used for inheritance purposes, because it provides access to the largest family of inherited characteristics. Since the blade-object and lever-object come from distinct geometric primitives (any vs. wedge), both are listed.

Screwdriver

A simple straight bladed screwdriver is illustrated in Fig. 2.61.

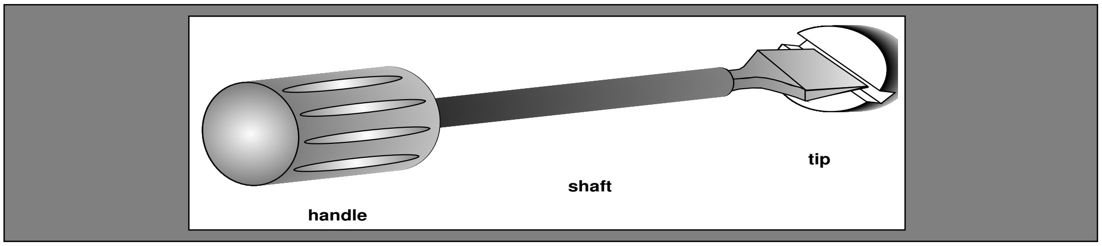


Figure 2.61 Screwdriver illustration.

The screwdriver has three components, a handle, a shaft, and a tip. These components are idealized with a wheel-object, a linkage-object, and a blade-object, as shown in Fig. 2.62.

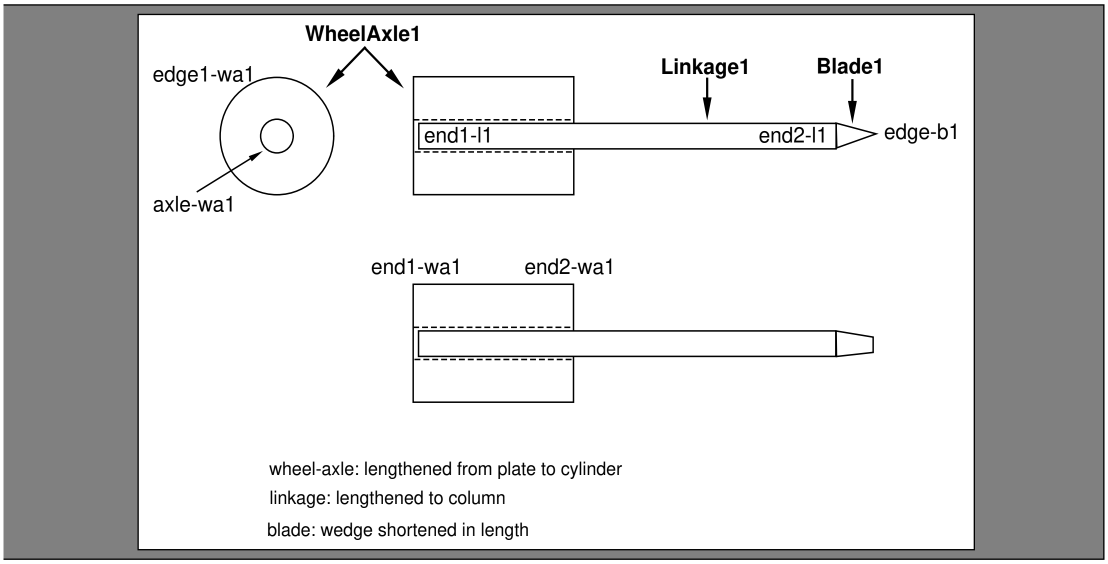


Figure 2.62 Screwdriver object primitives. A wheel-object, a linkage-object, and a blade-object.

The SDD for the screwdriver is shown in Fig. 2.63.

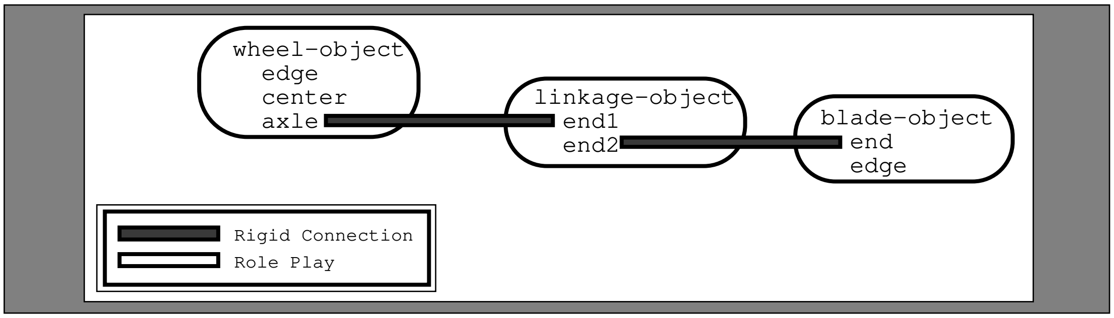
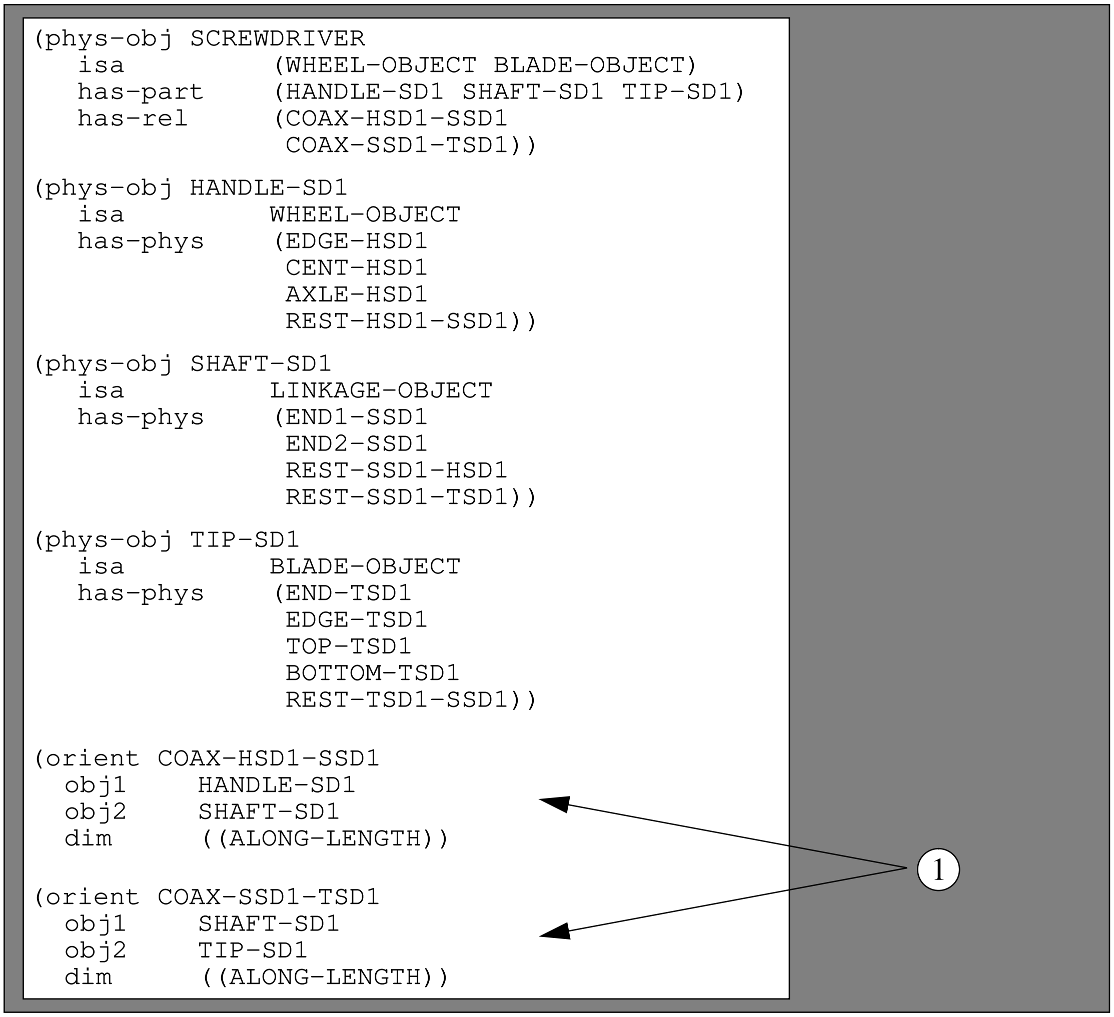


Figure 2.63 Screwdriver static representation diagram.

Like the other devices in this section, the screwdriver components are rigidly connected. Like the bottle opener, the screwdriver components are colinear, and, in the screwdriver, they are also coaxial, as represented (at 1) in Fig. 2.64.



```
(phys-obj SCREWDRIVER
   isa         (WHEEL-OBJECT BLADE-OBJECT)
   has-part    (HANDLE-SD1 SHAFT-SD1 TIP-SD1)
   has-rel     (COAX-HSD1-SSD1
                COAX-SSD1-TSD1))

(phys-obj HANDLE-SD1
   isa         WHEEL-OBJECT
   has-phys    (EDGE-HSD1
                CENT-HSD1
                AXLE-HSD1
                REST-HSD1-SSD1))

(phys-obj SHAFT-SD1
   isa         LINKAGE-OBJECT
   has-phys    (END1-SSD1
                END2-SSD1
                REST-SSD1-HSD1
                REST-SSD1-TSD1))

(phys-obj TIP-SD1
   isa         BLADE-OBJECT
   has-phys    (END-TSD1
                EDGE-TSD1
                TOP-TSD1
                BOTTOM-TSD1
                REST-TSD1-SSD1))


(orient COAX-HSD1-SSD1
  obj1    HANDLE-SD1
  obj2    SHAFT-SD1
  dim     ((ALONG-LENGTH))

(orient COAX-SSD1-TSD1
  obj1    SHAFT-SD1
  obj2    TIP-SD1
  dim     ((ALONG-LENGTH))
```

Figure 2.64 Representation of the coaxial component orientation of screwdriver components.

In this figure, the screwdriver components are related through their orientations and through their connections (although the connections are not represented in Fig. 2.64). The component regions have not been represented in this figure, as they simply point to the region equivalents of the wheel-object, the linkage-object, and the blade-object, respectively. Likewise, the restraint states have not been represented, since they are all locally rigid and have been represented elsewhere.

2.7.2 Simple Compound Devices

Simple compound devices have components which can move relative to one another. The complexity of the static description changes due to the inclusion of relational characteristics and bounding states of each component. In the following sections, two simple component devices are represented: (1) a nutcracker, and (2) a clothes pin. Three additional simple compound devices: (3) a pair of scissors, (4), a pair of pliers and a (5) door are represented in Appendix A.2

Nutcracker

The simple nutcracker (initially introduced to show device components, in Fig. 2.52) is pictured again in Fig. 2.65.

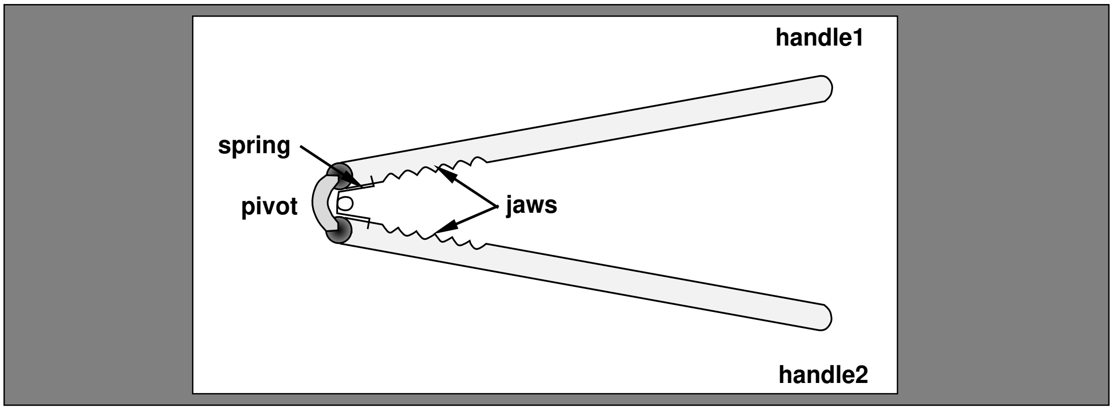


Figure 2.65 Nutcracker illustration.

Each lever component has three areas of interest: a place to hold and manipulate the device, a place to connect the two components, and a place where the force is produced and the object is held. The nutcracker components and the regions used to represent their functional areas are idealized as two lever-objects, a linkage-object, and a spring-object (Fig. 2.66). Fig. 2.66 shows two lever-objects where the location of the pivot and loc2 regions are reversed. The lever-objects are connected to the linkage object at their pivot regions, similar to the scissors and pliers devices. A spring is located at the same place and is in contact with the lever-objects. The jaw regions are idealized as the loc2 region for the lever-

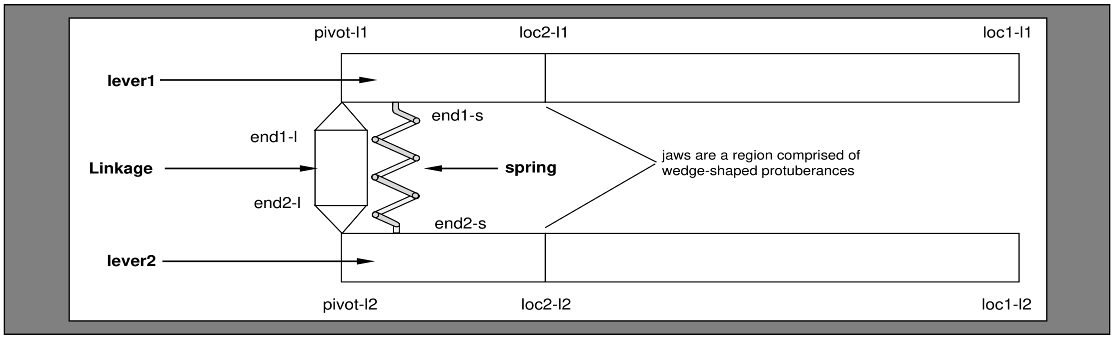


Figure 2.66 Nutcracker object primitives. Two lever-objects, a linkage-object, and a spring-object.

objects, but can be replaced by a more elaborate jaw as shown in Fig. 2.67.

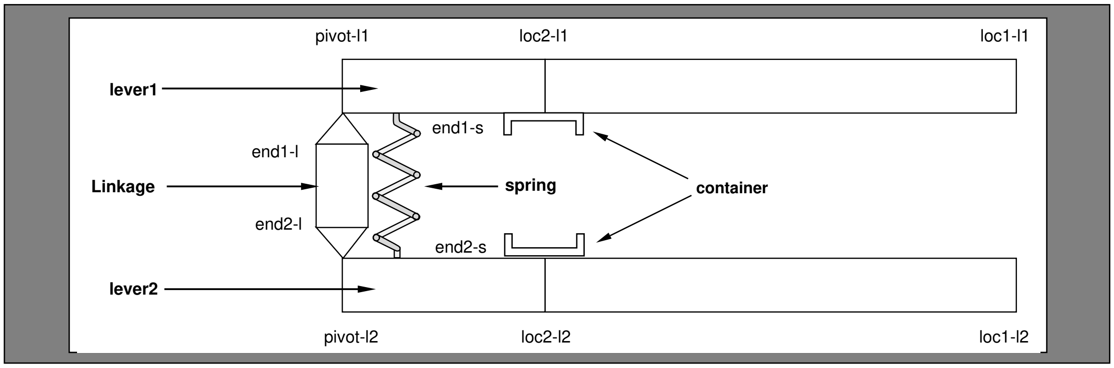


Figure 2.67 Modified idealization of nutcracker with jaw region represented with a container-object.

In this second version, the jaw is represented with a single container-object. A third version, which is shown as an idealized jaw, alone, is depicted in Fig. 2.68. Although these elaborations improve the ability to resolve the role of the jaw, the simple version in Fig. 2.66 represents the primary role of the jaw for constraining motion of a nut against the direction of applied force. The container and container/blade versions add to the representation the ability to represent motion constraint in more than one dimension, and the ability to represent holding ability through deformation, respectively, but they also increase the complexity of the representation. The accompanying SDD (Fig. 2.69) illustrates the connectivity of the object primitives in Fig. 2.66.

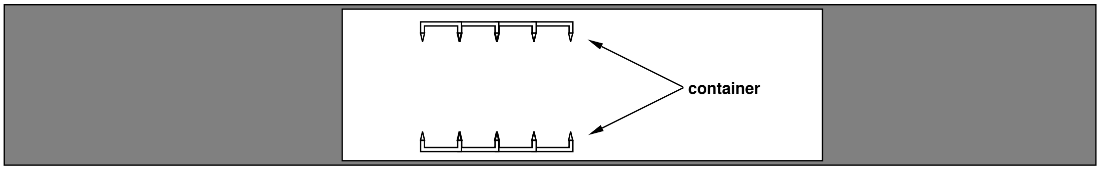


Figure 2.68 Jaw region represented as a series of container-object/blade-object pairs.

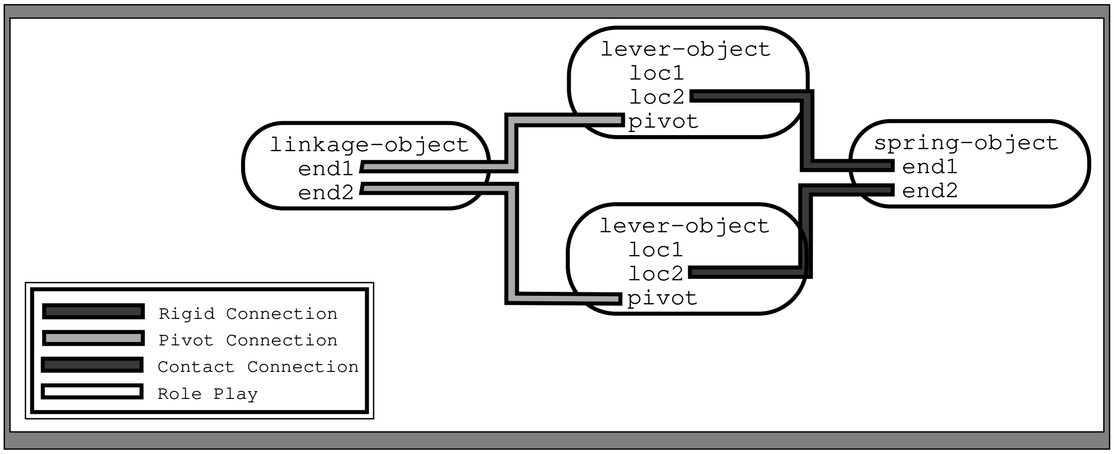


Figure 2.69 Nutcracker static representation diagram.

Two new connection arcs are shown in this figure, for: (1) contact, and (2) pivot. As previously mentioned, connection arcs depict a dynamic relationship between the components which gives rise to the restraint states associated with connectivity. The gray arcs between the linkage-object and the two lever-objects depicts a pivot relationship which allows relative rotational movement of either object. Likewise, the striped arcs between the lever-objects and the spring-object depicts a simple contact between the components.

Clothes Pin

A clothes pin is depicted in Fig. 2.70. The clothes pin is similar to a nutcracker inasmuch as they both have two handles and a spring. When the handles of either device are pressed together, there is resistance provided by the spring. To this extent, then, the static representations of the nutcracker and clothes pin should be similar if not identical. However, the spring is located at a different handle region, and the clothes pin actually works differently, so a dynamic representation should show differences which do not show up in the static representation. Fig. 2.71 shows an idealized version of the clothes pin. The device is represented with two lever objects and a spring-object as with the nutcracker. The positions of the pivot and jaw regions are swapped: the clothes pin "jaw" or clamp is located where the nutcracker pivot was, and the clothes pin pivot is located where the nutcracker jaw was. An idealized version of the clothes pin is illustrated in Fig. 2.71. Unlike the nutcracker, the clothes pin does not have a pivot component. The spring plays the role of the pivot and the

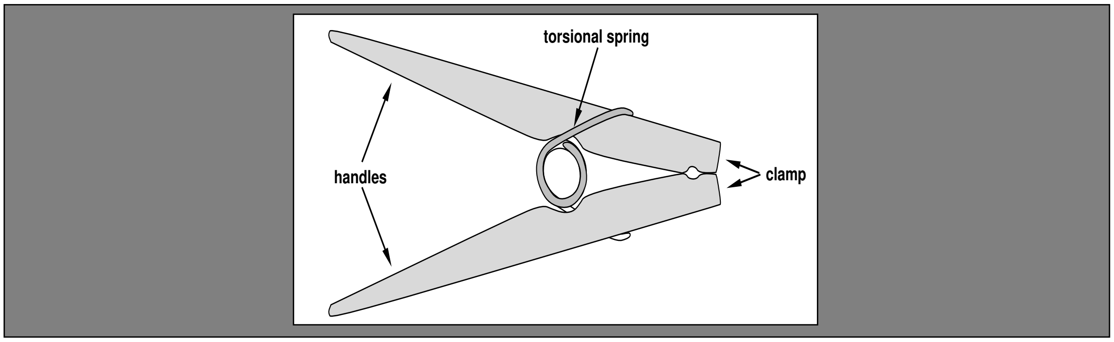


Figure 2.70 Clothespin illustration.

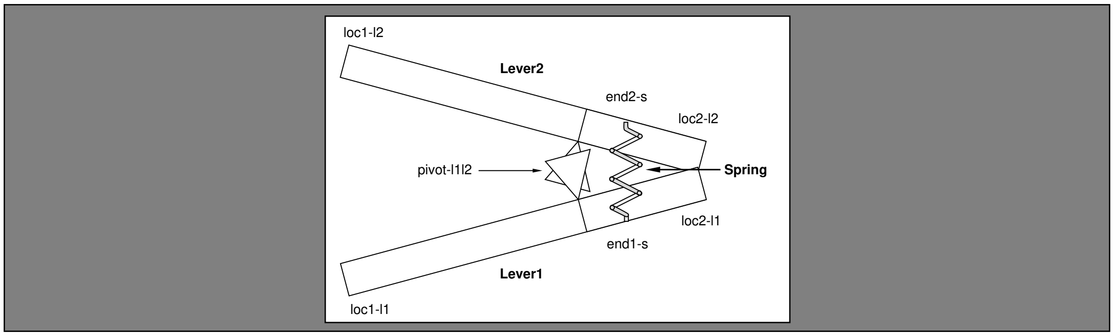


Figure 2.71 Clothes pin object primitives. Two lever-objects and a spring-object.

spring, so the SDD illustrated in Fig. 2.72 is commensurately simplified.

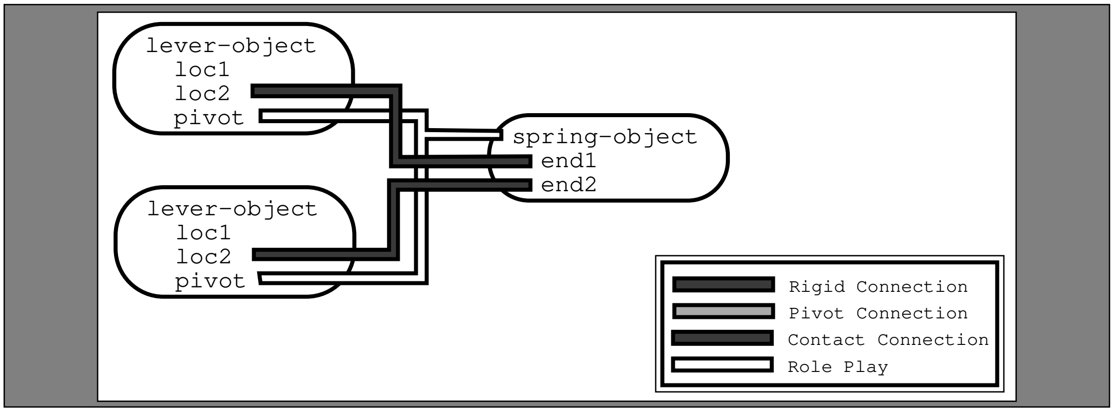


Figure 2.72 Clothes pin static representation diagram.

A new connection arc is presented in this figure. Since the spring actually plays the role of a linkage, a role-play arc is used to represent the associated connectivity, since there is really no connectivity between the spring-object and the lever-objects which isn't captured by the contact arc. The endpoints of the role-play arc are the regions of each component which are identical.

2.7.3 Multiple Compound Devices

A multiple compound device has multiple interacting components and multiple primitive object categories. Not only can components move relative to one another, they can do so in parallel. A component of a multiple component device may instantiate more than one type of device and the representation must capture this possibility. One multiple component device, a fingernail clipper, is represented in this section. Two additional multiple compound devices: (2) a press, and (3) a crank can opener, are represented in Appendix A.3.

Fingernail Clipper

A fingernail clipper is shown in Fig. 2.73. The device has four physically distinct components: (1) a handle, (2) a pin, and (3)(4) two cutters. In this device, the cutters play the role of blade-objects, of spring-objects, and of linkage-objects. An idealized version of the fingernail clipper is depicted in Fig. 2.74. In this figure, the separate functions of the individual components are fleshed out as unique object primitives, so there are seven idealized

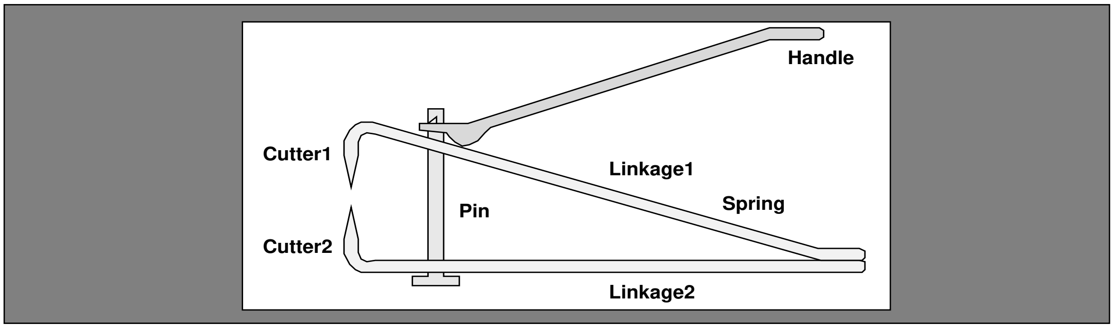


Figure 2.73 Fingernail clipper illustration.

objects used to represent the device rather than four. The fingernail clipper is very similar

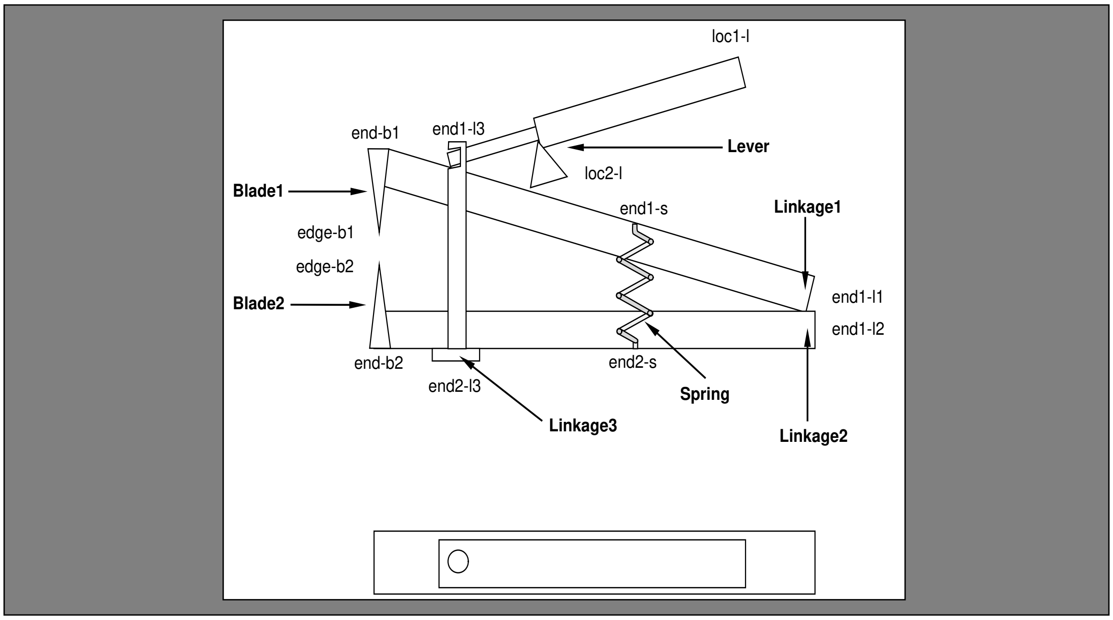


Figure 2.74 Fingernail clipper object primitives. A lever-object, a spring-object, three linkage-objects, and two blade-objects.

to a nutcracker and to a clothes pin. The device is similar to a clothes pin because the spring is not really at the end but in the middle where the force is applied. The device is similar to a nutcracker because the linkage-objects pivot at the end. The device is dissimilar to both in that force is applied at some intermediate region. These similarities and differences must be made explicit in both the SDD and the representation in general. The fingernail clipper SDD is shown in Fig. 2.75. The lower right-hand portion of the figure (at 1) shows a strong similarity to the SDD for the clothes pin, except that instead of a lever-object being in contact we have a linkage object and role-play relationships. Note that there is a rigid connection represented between the ends of the linkage-objects instead of the pivot connection between the handles of the nutcracker. The components have been arranged in a similar pattern to their physical appearance in Figs. 2.73, 2.74 to aid in following the connectivity between object primitives. The handle is represented with a lever-object (at 2). The region where force is applied, `loc2`, is represented as having a contact relationship with the upper linkage-object (3), but a region of contact is not specified. The pivot region of the lever-object is shown with a pivoting connection with the linkage-object at (5), which represents the fingernail clipper pin. This same linkage-object is also in contact with the lower linkage-object. The two blade-objects are depicted rigidly connected with the linkage-objects (4).

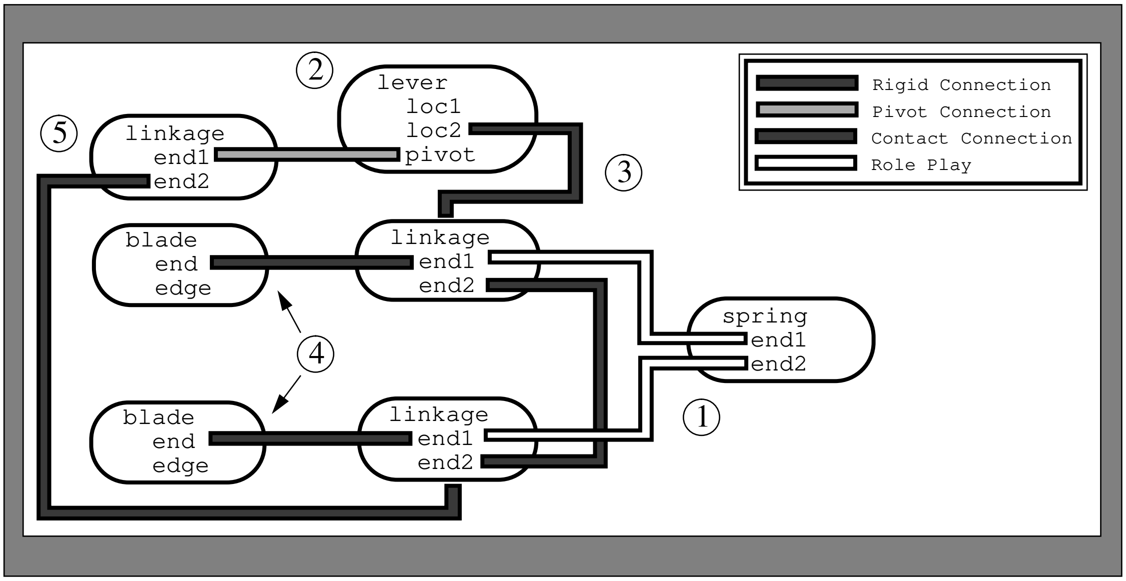


Figure 2.75 Fingernail clipper static representation diagram.

2.7.4 Complex Devices

Complex devices increase the number and type of components to the device, often requiring subgroups or mechanisms which perform specific tasks. One such device, a toy gun, is discussed and represented in this section. Three additional complex devices: (2) a corkscrew, (3) an eggbeater, and (4) a mousetrap are represented in Appendix A.4.

Toy Gun

The toy dart gun is a device with four major components: (1) a dart, (2) a body, (3) a trigger, and (4) a spring. The spring is single component, the dart and body are simple devices, and the trigger is a simple compound device. Unlike other devices mentioned in this text, the toy dart gun has a component which completely separates from the device in normal operation. The complexity of this device is seen in the idealized toy gun depicted in 2.77. In this figure, the dart is comprised of three linkage-objects of different sizes. The gun body is idealized as a container-object and a linkage-object, and the trigger mechanism is represented as a lever-object. The accompanying SDD depicts the dart as located in the barrel of the toy gun (1). One end of the linkage-object, `end2`, is represented as being in contact with the spring-object `end1` region and `inside` region of the container-object (2), and with the

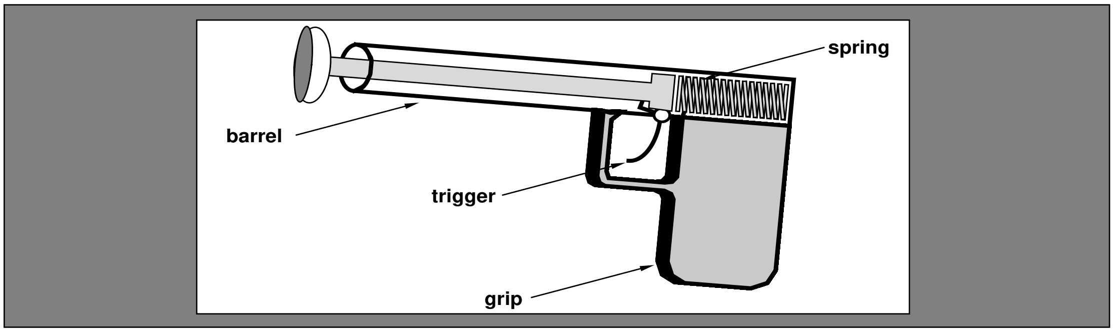


Figure 2.76 Toy gun illustration.

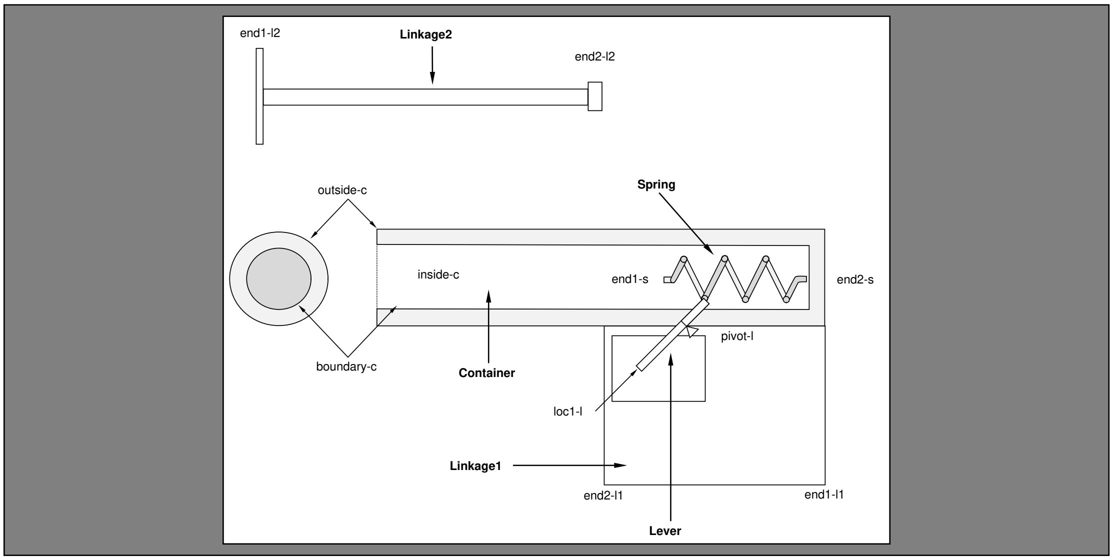


Figure 2.77 Toy gun object primitives. A container, a linkage, a spring, and a lever.

`loc2` region of the lever-object which represents the trigger (3). The lever-object pivot region has a pivot connection with the `outside` region of the container-object. The container object `outside` region is also rigidly connected to the linkage-object (5). Finally, the other spring-object end region, `end2`, is in contact with the `inside` region of the container-object (6).

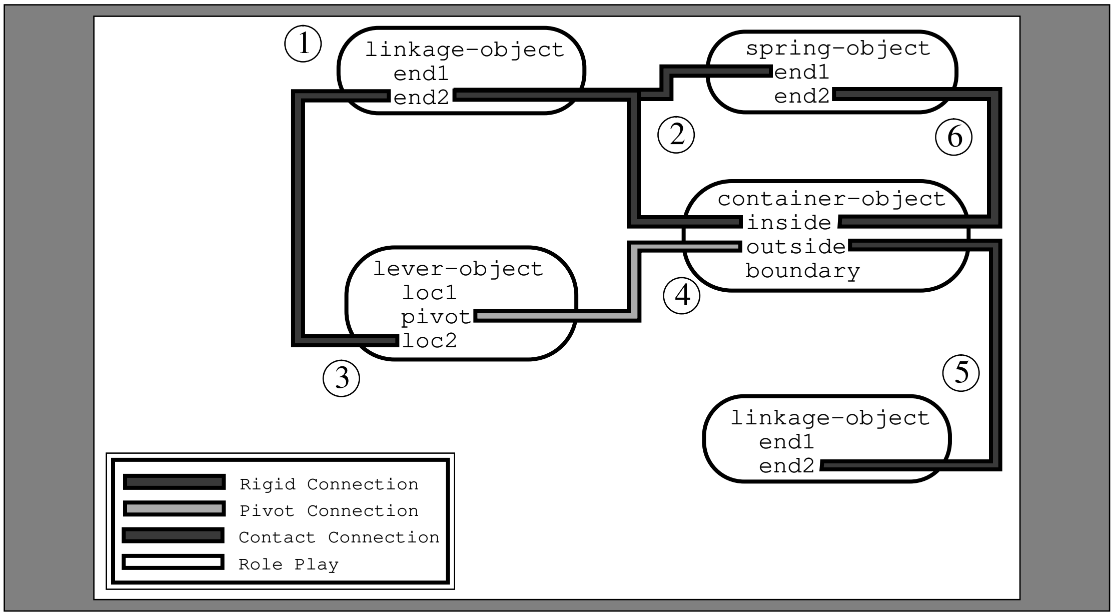
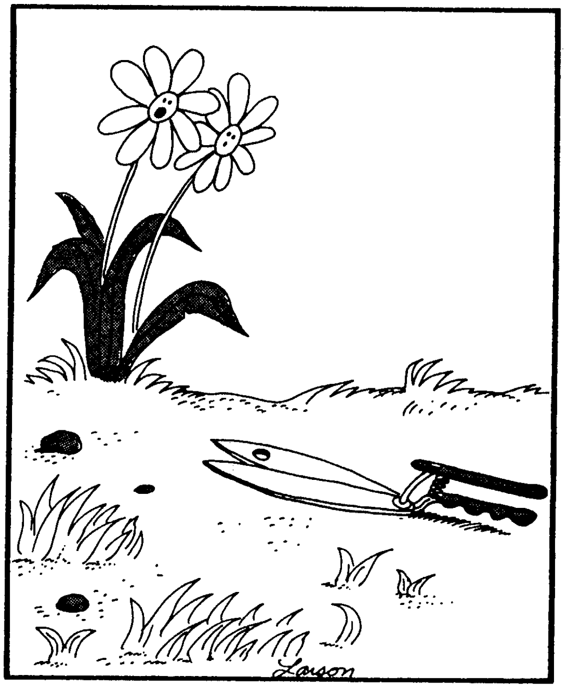


Figure 2.78 Static representation diagram for toy gun.



"There! There! See it, Larry? . . . It moved a little closer!"

○
○
○
○

Chapter 3

Low-Level Device Dynamics: Behavior

Devices are used to perform problem solving tasks by undergoing change or by producing change in other objects in the environment. When a force is applied to an object, the object is said to be *perturbed* and, depending on its static description, may undergo state changes. Changes in state describe object *behavior*. For example, if an object is unrestrained in a dimension and is then forced in that dimension, then the object will change position. Because of this, object position is called a behavioral property. Every device function can be described as a combination of mechanical behaviors.

There are two ways of describing device behavior: (1) by what a device does, and (2) by how it does it. What a device does is called a *function*. Device function describes what perturbations are applied to a device and what happens as a result of the perturbation. This is called global or *high-level* behavior. Component behaviors and their combinations are called local or *low-level* behavior. Both high-level and low-level behavior describe physical change at different levels of granularity. The combination of low-level and high-level mechanical behaviors describes how objects behave and how the interactions between objects are perceived. In FONM, the causal dependencies which are used to represent low-level and high-level device behavior are called *device dynamics*. An object can be used in a specific task if its dynamic description produces the same states that its intentional description does under the same conditions.

When a device is perturbed in some way, the manner in which the perturbation is propagated throughout the device, and how it affects other objects, is dependent on the combined behavior of its components. For example, the nutcracker illustrated in Fig. 2.52 can be used to crack nuts open by applying leverage. When the user pulls the handles together, a force is produced in the handles. This force is transmitted through the pivoted ends, to the pivot component, and to the pecan. Because of the pivoted connection, the handles are able to rotate, but the pecan constrains their motion. The relative distance between the handles, the pivot location, and the pecan contact location magnify the force to an extent that its value exceeds the breaking strength of the pecan shell, and it cracks open. These interactions: rotation, pivot, transmission, magnification, and cracking describe behaviors which must be represented in order to describe complex mechanical behavior and function. As a result, mechanical behavior is the basic dynamic component in the FONM approach, because representing and recognizing behavior enables process models which use FONM representations to predict and explain object perturbations and their causal dependencies.

In this chapter, the representation constructs which are used in FONM to describe low-level device dynamics are presented. All device behavior can be described as permutations in the static descriptions of device components, so the discussion begins with a description of permutations which can be described for static object descriptions. The discussion ends with the presentation of device dynamics for six mechanical devices of increasing complexity. Three topics are addressed in this chapter:

(1) Low-level object behavior
(2) Behavioral processes
(3) Process sequences and high-level object behavior

3.1 Low-Level Object Behavior

Each aspect of the object's static description affects its behavior. The behavioral properties describe what behaviors can take place, such as knowing an object's position, which is important in determining whether it is in contact with another object. The object's material properties causally enable or disable behaviors in which it might engage. For example, it is reasonable to assume that a steel knife can be used to whittle wood and that a rubber one cannot. To make this prediction, the dynamic relationship between material stiffness/strength and cutting must be made explicit. That is, FONM must represent that steel being harder than wood and rubber being softer than wood are causally related to cutting. Finally, the object's regions determine where behavior will take place, and the connections to other objects determine where behaviors will be propagated. For example, if it is known that two regions share the same location, then force can be transmitted between them at that location.

Each of the five FONM behavioral states is used to represent changes a mechanical object may undergo. A sequence of any particular behavioral property states for an object is called a *history* for the property, and the physical interaction responsible for the state change is called a behavioral *process*. A process and state history describe an object's low-level behavior for a particular property.

3.1.1 Continuous and Discrete Models of Mechanical Behavior

In FONM low-level physical behavior is represented qualitatively; as discrete models of continuous behavior. Mechanical objects move and stop, and they store, transmit, and transform energy. In so doing, they may transform their own physical form or that of other objects. In Applied Mechanics, these physical interactions are described as continuous behavioral processes. In qualitative physics, each of these interactions is considered to be qualitatively different; each describes a unique behavioral process. The difference is illustrated with the following example. Consider a spring stretching and a paperclip bending. In Applied Mechanics, stretching and bending are both described by a material's stress-strain characteristics in a theory of bending. Any object strain below the elastic limit describes stretching, while strain beyond the elastic limit describes bending. Stretched objects will, when released, return to their original size and shape. Bent objects, when released, retain some of their modified size and shape. In FONM, stretching and bending result in different behavioral property states, so they are represented with different behavioral processes. Instead of a single theory which describes all the behavioral dependencies, the FONM approach is to have a theory for each process and describe the dependencies between those processes.

3.1.2 Process Consistency with Applied Mechanics

A discrete model of device dynamics is only as good as its agreement with the quantitative models of Applied Mechanics. There are three conditions under which the discretization of physical behavior into behavioral processes can be considered consistent with Applied Mechanics, namely when they are: (1) dynamically consistent, (2) predictively consistent, and (3) continuity consistent. Behavioral processes are *dynamically* consistent with Applied Mechanics if their behavior is based on the same principles upon which Applied Mechanics is based. FONM processes are based on a restraint model. Objects which are not otherwise specified are free to participate in processes, and processes which are enabled remain enabled unless otherwise acted upon. For example, if an object is not specifically restrained in a dimension and direction, then it is free to move in that dimension and direction. Unless other knowledge is specified, a generic object is free to move in any dimension and could be recognized as a falling object. The same object, once falling, will continue to fall unless it collides with an object or is otherwise acted upon by another process.

Behavioral processes are *predictively* consistent with Applied Mechanics if, given the same object and conditions, predictions based on qualitative processes agree with those based on Applied Mechanics. Consider an object which is stretched beyond its elastic limit. If the object is initially recognized as a participant in a stretching process, and then as a participant in a bending process, and the resulting states qualitatively agree with the stress-strain analysis predicted by Applied Mechanics, then the model is consistent for this example. Predictive consistency must hold regardless of the representational granularity. For example, it doesn't matter how a game of catch is represented as long as those ball physical states which are qualitatively described by the model agree with those predicted by Applied Mechanics.

Behavioral processes are *continuity* consistent with Applied Mechanics if they describe continuous behavior when Applied Mechanics would describe continuous behavior. This condition guarantees that transient behavior, which is ignored in FONM behavioral processes, does not produce discontinuities unless those discontinuities would be produced in a continuous model. For example, a stretching process results in a size state change, and an internal force state change on the stretched object. If these states collectively enable the bending process, then the two discrete processes are continuity consistent with Applied Mechanics.

3.2 Behavioral Processes

Behavioral processes describe dynamic behavior: they are enabled and disabled by physical states, and they result in physical state changes. In FONM, behavioral processes are represented with the process knowledge structure. Processes have six roles: src, dst, dimr, from, to, and inst. The *src* and *dst* roles refer to the source and destination locations (regions or objects) where the interaction takes place. The *dimr* role refers to the applicable dimension and direction of the process. The *from* and *to* roles refer to enabling, and resulting states of the process, respectively, and the *inst* role refers to an object or region whose function mediates the process. There are two kinds of process enablement, as shown in Fig. 3.1: behavioral preconditions and compositional preconditions. *Behavioral* preconditions (1) are behavioral property states and take the same form as the resulting state (2).

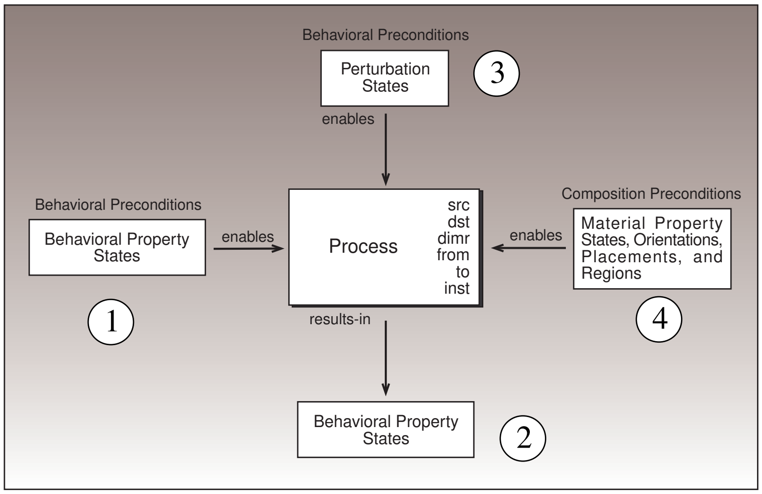


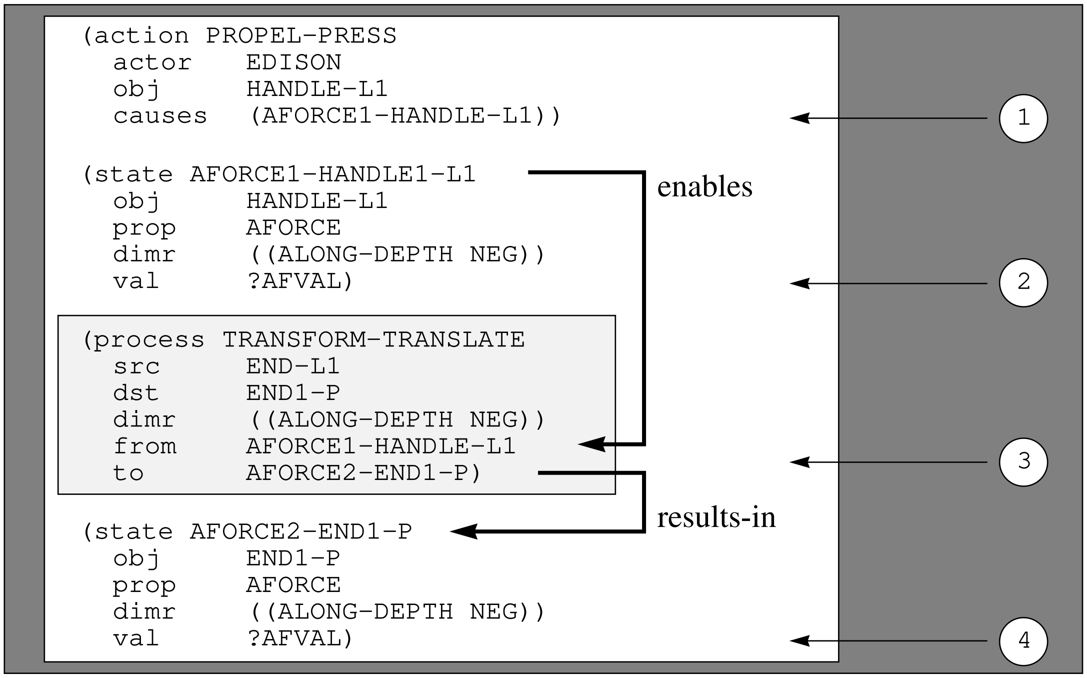
Figure 3.1 Process enablements and resulting states.

Perturbation states (3) are aforce states, and are set apart in Fig. 3.1 because they can result from actions (such as PROPEL-PUSH) as well as from processes which effect aforce state changes. *Compositional* preconditions (4) are material property states. In the rubber knife example discussed above, the material stiffness property value is an enablement for cutting. When specific property values are associated with object behavior, they enter the representation as compositional preconditions.

Object behavior has both local and global effects. Locally, each object region can participate in processes, resulting in local behavior states. The region then fills the *src* or *dst* process roles, and the *to* role states resulting from an enabled process are *local* states associated with the region at which the process takes place. The union of object local states describes a *global* state for the entire object and property. For example, the union of local aforce states on an object describes the global object aforce state.

When the process src and dst roles are filled with objects, rather than regions, the object CG is assumed and no local behavior can be inferred, because the locations where the behavior takes place are not specified.

The process representation can be statically illustrated by considering the action of squeezing the two nutcracker handles together in the absence of a nut (Fig. 3.2). The transmission of force from each linkage to the pivot piece is represented using a process which describes force transformations, called TRANSFORM.



```
(action PROPEL-PRESS
  actor   EDISON
  obj     HANDLE-L1
  causes  (AFORCE1-HANDLE-L1))

(state AFORCE1-HANDLE1-L1
  obj     HANDLE-L1
  prop    AFORCE
  dimr    ((ALONG-DEPTH NEG))
  val     ?AFVAL)

(process TRANSFORM-TRANSLATE
  src     END-L1
  dst     END1-P
  dimr    ((ALONG-DEPTH NEG))
  from    AFORCE1-HANDLE-L1
  to      AFORCE2-END1-P)

(state AFORCE2-END1-P
  obj     END1-P
  prop    AFORCE
  dimr    ((ALONG-DEPTH NEG))
  val     ?AFVAL)
```

Figure 3.2 Process representation for force transmission.

Fig. 3.2 represents the states which enable the translation of force applied at the nutcracker handle, through the pivoted connection with the pivot component, to the pivot end. In the example, the nutcracker components LEVER1 (L1) and PIVOT (P) are assumed to be in contact in the dimension and direction (ALONG-DEPTH NEG). The lever handle, HANDLE-L1, when pressed (1), causes an applied force state at the same location and in the same (ALONG-DEPTH NEG) dimension and direction. In FONM, when a force is applied to an object, the force is conveyed to all regions. That is, there are no transient effects modeled in FONM. The aforce value is divided into the number of connections which can support force transmission in the specified dimension and direction. Thus the aforce at HANDLE-L1 (2) has the same value (AFVAL) as that at the region END-L1, because it is LEVER1's only regional location which is connected to PIVOT. The *src* role of TRANSFORM-TRANSLATE is filled by the region HANDLE-L1. The *dst* role is similarly filled by the region END1-P. The *from r*ole is filled by a state (3) which represents the LEVER1 force in dimr (ALONG-DEPTH NEG) prior to the process, and the *to r*ole is filled by a state which represents the force transmitted by HANDLE-L1 to the pivot piece (4). When the same interactions are described for LEVER2 and PIVOT, a force of the same magnitude is transmitted to END2-P, but in the (POS) direction. When the global aforce state on PIVOT is updated, these two forces cancel one another and the representation produces the same result as a nutcracker in static equilibrium.

3.2.1 Behavioral Process Primitives

TRANSFORM is called a behavioral process primitive (BPP). In FONM, there are five BPPs, distinguished by the type of behavioral property state which they affect: MOTION, RESTRAIN, TRANSFORM, STORE, and DEFORM. These primitives form a causal hierarchy for describing mechanical behavior, because the state changes which they represent describe the range of possible mechanical behavior, as illustrated in Fig. 3.3 and Table 3.1:.

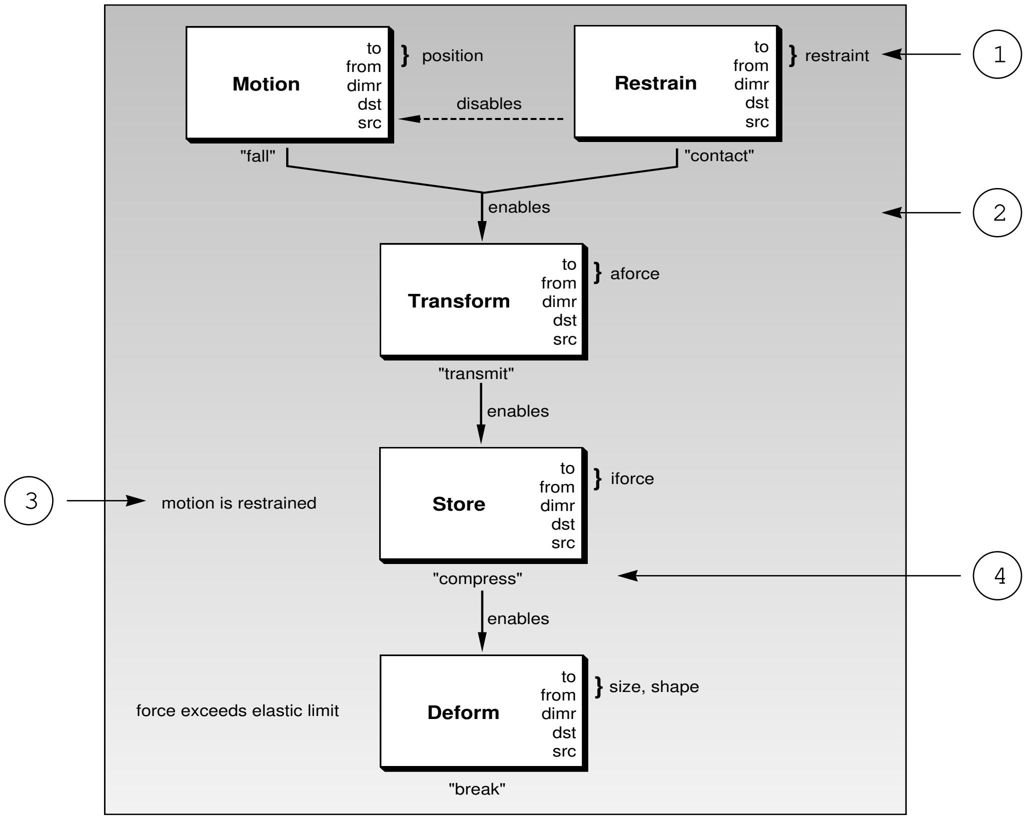


Figure 3.3 Behavioral process primitives.

In Fig. 3.3, each primitive class is depicted as a process. The behavioral property state type resulting from an enabled primitive is labeled outside the right side of each process (e.g., restraint at 1). The causal dependencies between primitive classes are depicted on the arcs connecting the processes (2). Representative enabling preconditions are shown (where appropriate) to the left of each primitive (3). Beneath each primitive is an example of a lexical specialization for each primitive (at 4, for reference).

Each BPP is represented with a process knowledge structure. MOTION describes object motion, and represents changes in object position. RESTRAIN describes object connectivity, and represents changes in object restraint. TRANSFORM describes force transformation between objects, and represents changes in applied force dimension, direction, and magnitude. In Applied Mechanics, the field of rigid-body statics describes force transformations which do not modify the object. TRANSFORM therefore describes qualitative rigid-body statics. TRANSFORM is enabled by an applied force on one object which (minimally) comes into contact with a second object. STORE describes non-permanent (elastic) deformation, and represents changes in internal force (energy). STORE is enabled by an applied force, TRANSFORM (nominally TRANSFORM-TRANSMIT), and a restraint state on the objects in contact. DEFORM describes permanent object deformation, and represents changes in physical size or shape. When an object is deformed elastically, such that its original size and shape are retained, STORE is implied. If the elastic limit is exceeded, for example when an object is bent or broken, DEFORM is implied.

Table 3.1: illustrates the relationships between BPPs, their behavioral and quantity pre-

conditions, and the physical states they cause.

Process Type	Behavioral Preconditions	Perturbation Preconditions	Resulting State
RESTRAIN	none	equivalent region position	restraint
MOTION	RESTRAIN (disables)	nonzero global force	position
TRANSFORM	RESTRAIN, MOTION	none	aforce
STORE	TRANSFORM	aforce < elastic limit	iforce
DEFORM	STORE	aforce > elastic limit	size

Table 3.1: Behavioral process primitives.

In FONM, process representations are always depicted as unidirectional. However, behavioral processes are bidirectional. For example, when an object is in motion its center of gravity will change position, but behaviorally there may be no difference between the object changing position and the object CG remaining stationary and everything else moving. The FONM representation implies directionality for the purpose of continuity, but the directionality is not intended to preclude the alternative point of view.

3.2.2 MOTION

Each BPP primitive represents a generic process class, and specializations on its roles represent various process subclasses. A *generic* process defines the minimum structure whereby the process can be recognized. For example, object motion is represented with the generic process MOTION, which is based solely on two behavioral preconditions: global restraint freedom in a dimension/direction, and global unbalanced force in the same dimension/direction. Once enabled for a particular object and dimension/direction, MOTION represents the resulting change in object position in dimension/direction. These relationships are illustrated in Fig. 3.4.

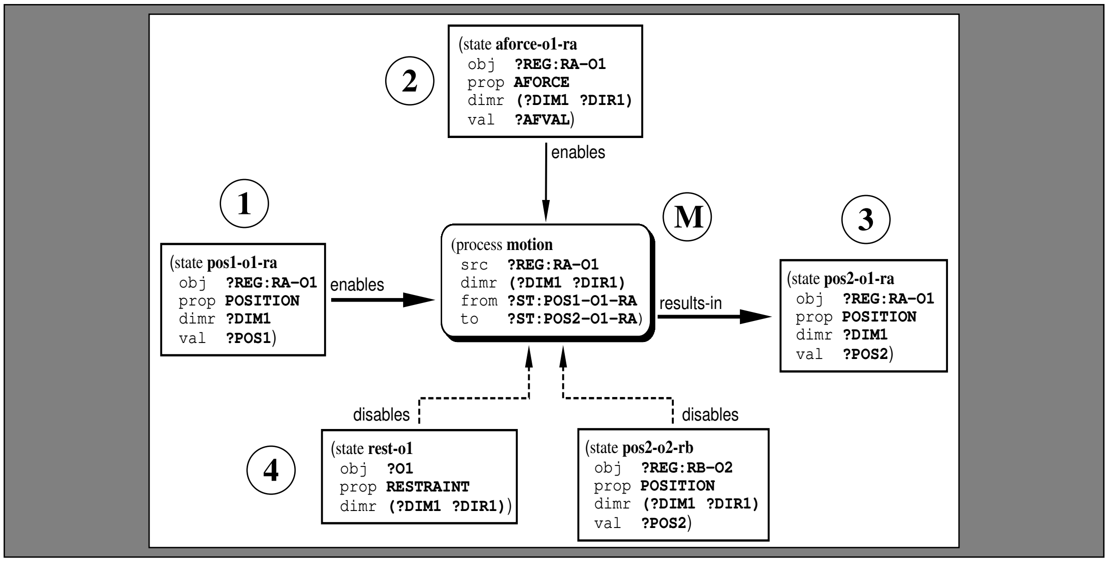


Figure 3.4 Frame representation for the MOTION behavioral process primitive.

Fig. 3.4 depicts the MOTION process as a schema (M) which takes an object position (1), and an applied force (2) and describes a change in position (3). The applied force defines the dimension and direction in which motion will occur, while the positions are defined by the dimension and a value. The original position state fills the BPP *from* role, while the new position state fills the BPP *to* role. The two disabling states (4) represent conditions which can disable the MOTION process. The first is a restraint state at the region where the force is applied, and the second is a second object which is positioned where the first will be when the process completes. The disablements are depicted with dotted lines to distinguish them from enabling conditions.

The generic MOTION process does not specify the object's path, only that the object's position will change and what the dimension will be. Motion specializations are represented by further specifying the type of applied force, the type and location of object restraints, or other physical property values. By specializing the MOTION enabling conditions, the process roles are specialized and path information can be described. Fig. 3.5 illustrates the family of MOTION class specializations which can be represented this way.

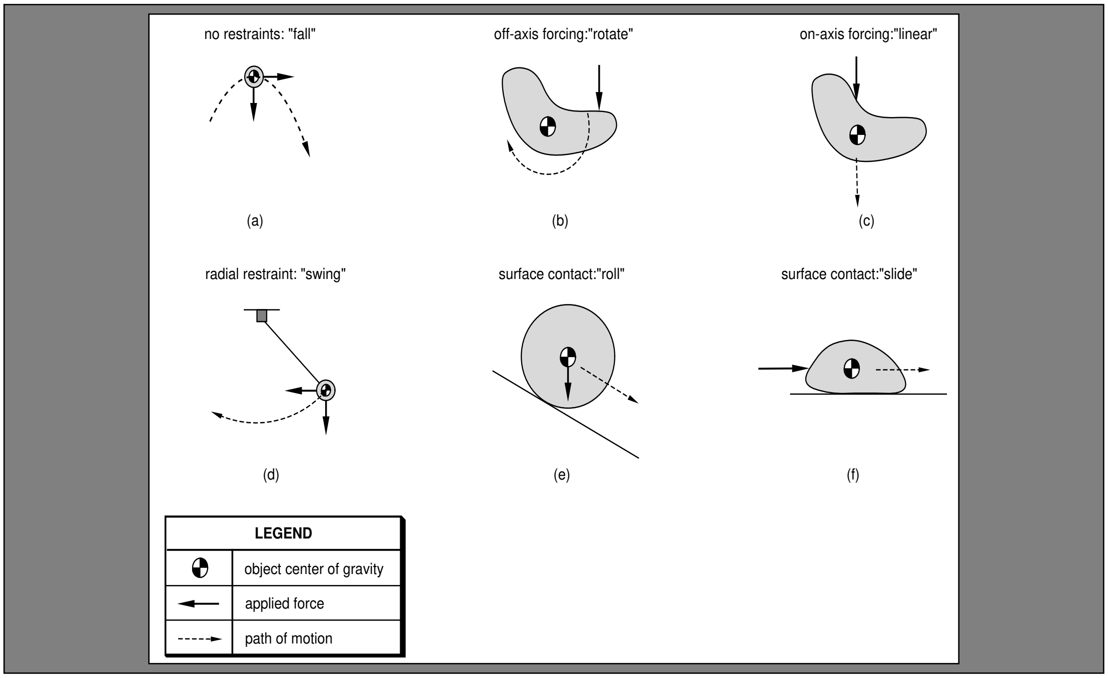


Figure 3.5 MOTION process specializations, (a) fall, (b) rotate, (c) linear, (d) swing, (e) roll, and (f) slide.

In Fig. 3.5, the dark arrows represent the location, dimension, and direction of applied force and the thin dotted arrows represent the path of motion. The motions depicted in Figs. 3.5a-f are representative of motions which MOTION should be able to describe. Fig. 3.5a illustrates MOTION-FALL, which represents an unrestrained object forced by a gravitational field. Figs. 3.5b and c illustrate rotational and linear motion as MOTION specializations where the forcing is off the centroidal axis dimension, and on the centroidal axis dimension, respectively. These three motion types are specialized to swinging, rolling, and sliding, Figs. 3.5d, e, and f by specifying a type of contact. The hierarchical relationships between generic MOTION and these six specializations are illustrated in Fig. 3.6 and tabulated in

Table 3.2.

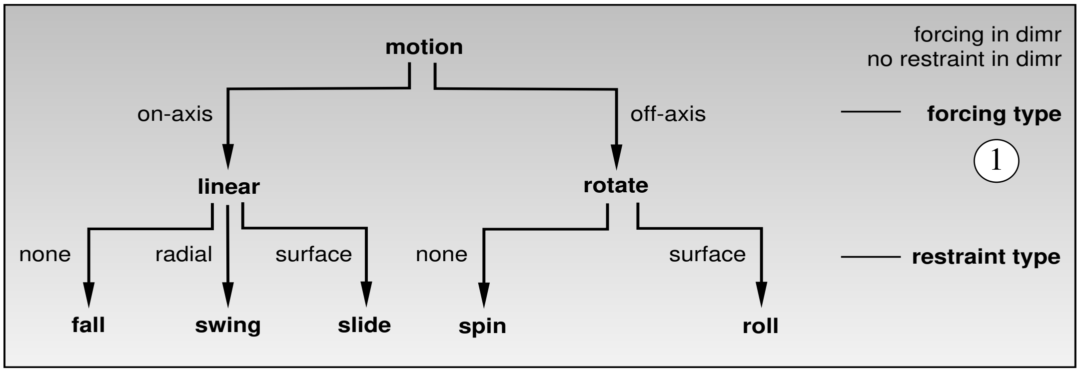


Figure 3.6 Hierarchical relationships between MOTION processes, differentiated by process role instantiations.

The hierarchy in Fig. 3.6 distinguishes motion processes based on how the MOTION behavioral preconditions are instantiated. The labels on the right hand side of the figure denote the precondition types (1). For example, by specifying whether or not the applied force is coaxial with the object CG, the motion class is distinguished between linear and rotation.

Motion Class	Unrestrained Dimensions Connection Type	Forcing Dimension	Motion Path
TRANSLATE	translation none	on-axis	straight
FALL	gravitational translation none	on-axis gravitational	vertical
SLIDE	translation surface contact	on-axis gravitational	planar
ROTATE	rotation pivot	off-axis	circular
SWING	gravitational translation hang	tangential gravitational	circular vertical
ROLL	translation, rotation surface contact	off-axis gravitational	circular

Table 3.2 Specializations of the MOTION behavioral process primitive.

Motion specializations are classified, individually, by their physical restraints and their forcing. Connection type is divided into translational freedom, rotational freedom, and surface contact. Translational and rotational freedom reference the dimension/direction of motion. Forcing type is divided into on-axis and off-axis forces. On-axis forcing is applied through the object's center of gravity, while off-axis forcing is applied elsewhere (as in tangential forcing). Motion types can be combined to describe complex paths. For example, an object rolling down a hill is both rotating and falling. The forcing for rolling is obtained by an off-axis component of gravitational acceleration, as illustrated in Fig. 3.5e.

The representation for MOTION in Fig. 3.4 is shown instantiated for the SLIDE motion specialization in Fig. 3.7. The perturbation is applied at region RA-O1, in dimension DIM1, and in direction toward the O1 CG (shown at 1).

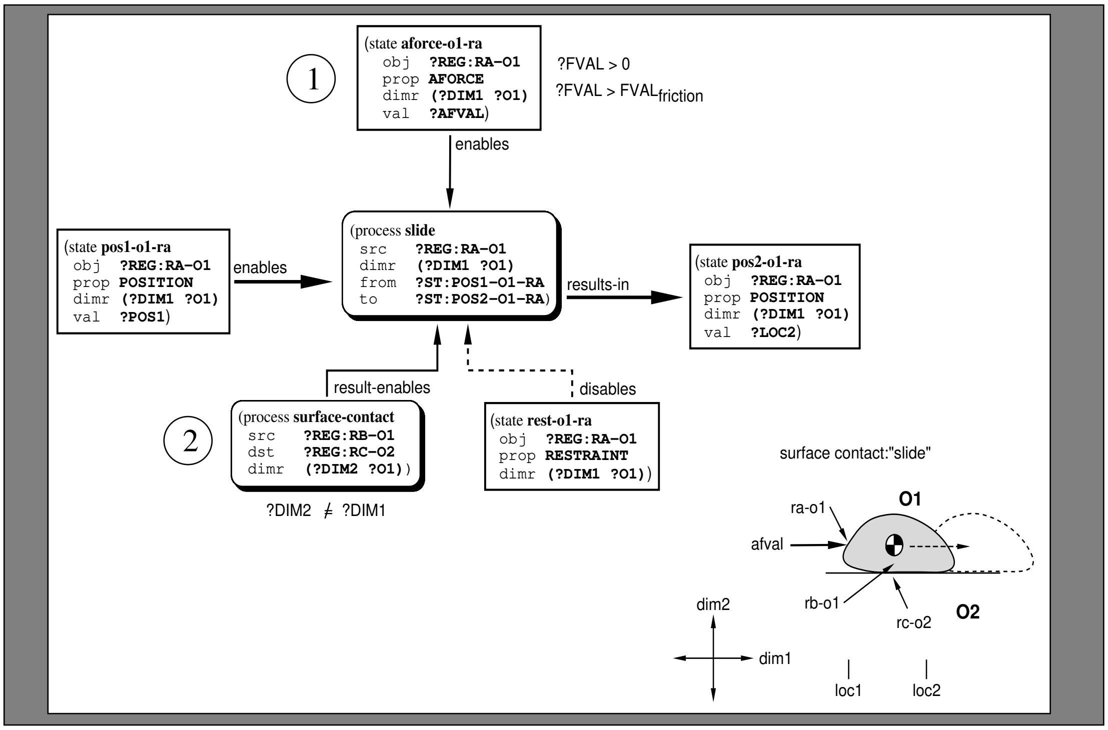


Figure 3.7 Graphical representation of SLIDE process specialization of the MOTION process class.

The value of the force state AFORCE-O1-RA, AFVAL, must be unbalanced to enable MOTION. To represent sliding with MOTION-SLIDE, AFVAL must exceed the friction force between object O1 and object O2. In addition, MOTION-SLIDE specifies surface contact with O2, and that the perturbation dimension, DIMR, not be the dimension of contact (2). These additional representational facets are shown at (1) and (2).

3.2.3 RESTRAIN

The relationships between objects in contact are often referred to as state relations, because they describe causal dependencies but do not appear to be dynamic. For example, when a book is lying on a table, it is being supported by the table, and "support" is considered a state relation between the book and table (i.e., the book is supported by the table). Although the observed result is a disabled potential for book motion, contact between the book and the table produces changes in the restraint states of each object: on the book a downward restraint, and on the table an upward restraint (see Fig. 3.8). It is the restraint states which disable the potential for book, and table, motion. These restraint states persist as long as the book and table remain in contact. When the contact is discontinued, the restraint states no longer exist. Thus described, "support" is not a state, per se, but a dynamic relation which describes a form of physical behavior.

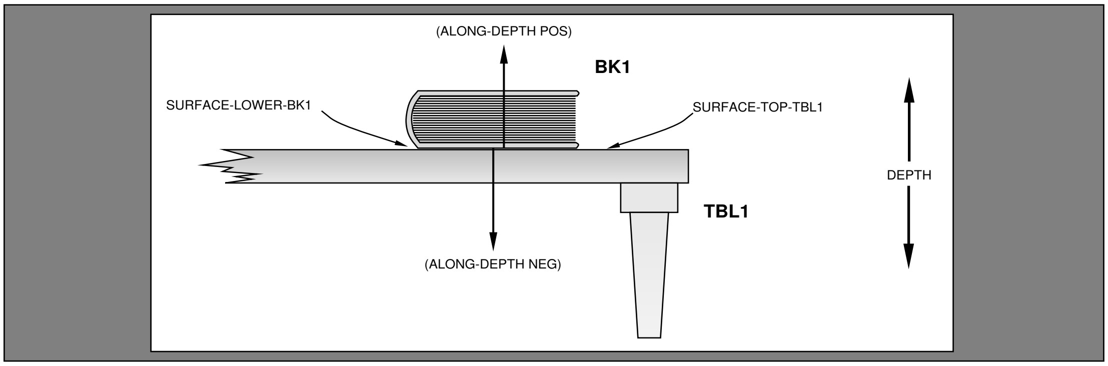


Figure 3.8 A book supported by a table describes restraint states on both the book and the table.

Fig. 3.8 illustrates the dependency between two objects in contact. The book cover, region SURFACE-LOWER-BK1, and the table top, region SURFACE-TOP-TBL1, share the same physical position in the vertical (ALONG-DEPTH) dimension. Each object has a local restraint state, in the vertical (ALONG-DEPTH) dimension, produced by the presence of the other object. The restraint state on the book is in the (ALONG-DEPTH NEG) dimension/direction, while the restraint state on the table is in the (ALONG-DEPTH POS) dimension/direction. According to the MOTION restraint disablement, BK1 motion is disabled in the (ALONG-DEPTH NEG) dimension/direction unless TBL1 also participates in MOTION in the same dimension/direction. The converse is also true; TBL1 cannot participate in MOTION in the (ALONG-DEPTH POS) dimension/direction without BK1.

In FONM, the RESTRAIN process describes objects in contact and the resulting restraint state changes. RESTRAIN involves two objects. Generic RESTRAIN, called CONTACT (Fig. 3.9), has a single behavioral precondition; each object has a region whose position is identical to that of the other object. The result of an enabled RESTRAIN is two local restraint states: one on each object, in the same dimension but opposite directions. RESTRAIN is a process, so the restraint states resulting from an enabled RESTRAIN remain active until the process is explicitly disabled.

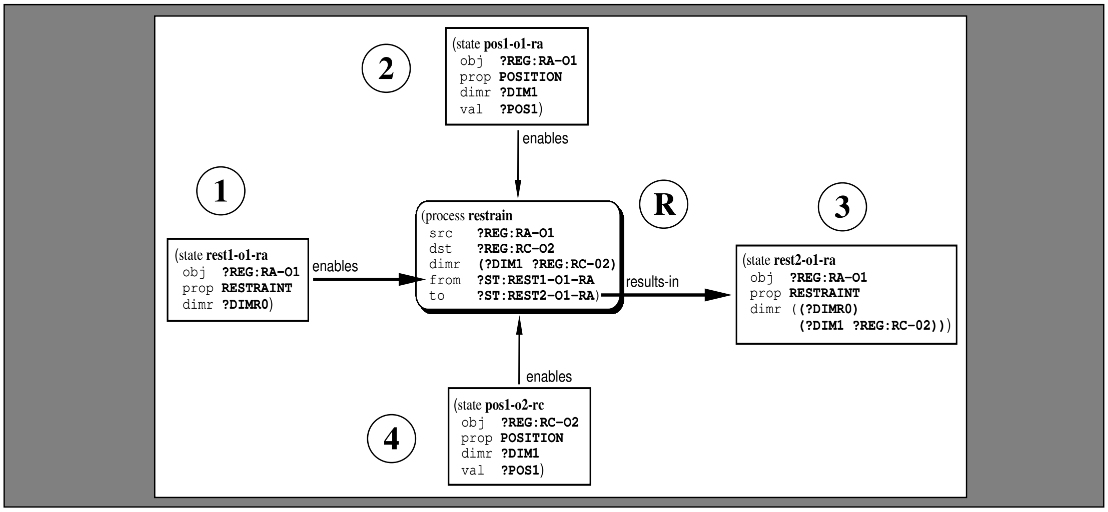


Figure 3.9 Frame representation for the RESTRAIN behavioral process primitive.

Fig. 3.9 shows how RESTRAIN organizes the interaction between restraint and position for objects in contact (R). The enabling condition, two object regions in the same location, is shown by the position states at (2) and (4), where the common position is POS1. The initial restraint state is labeled at (1) and the resulting state is labeled at (3).

Many contact relationships can be represented by specializing the RESTRAIN process. For example, the RESTRAIN process used to represent the book and table interaction is called SUPPORT. SUPPORT is a specialization of RESTRAIN-CONTACT where one object is placed above the other (i.e., in the vertical dimension). The opposite case, where one object is placed below the other is shown in Fig. 3.10, along with two other conditions which can be represented as RESTRAIN class specializations.

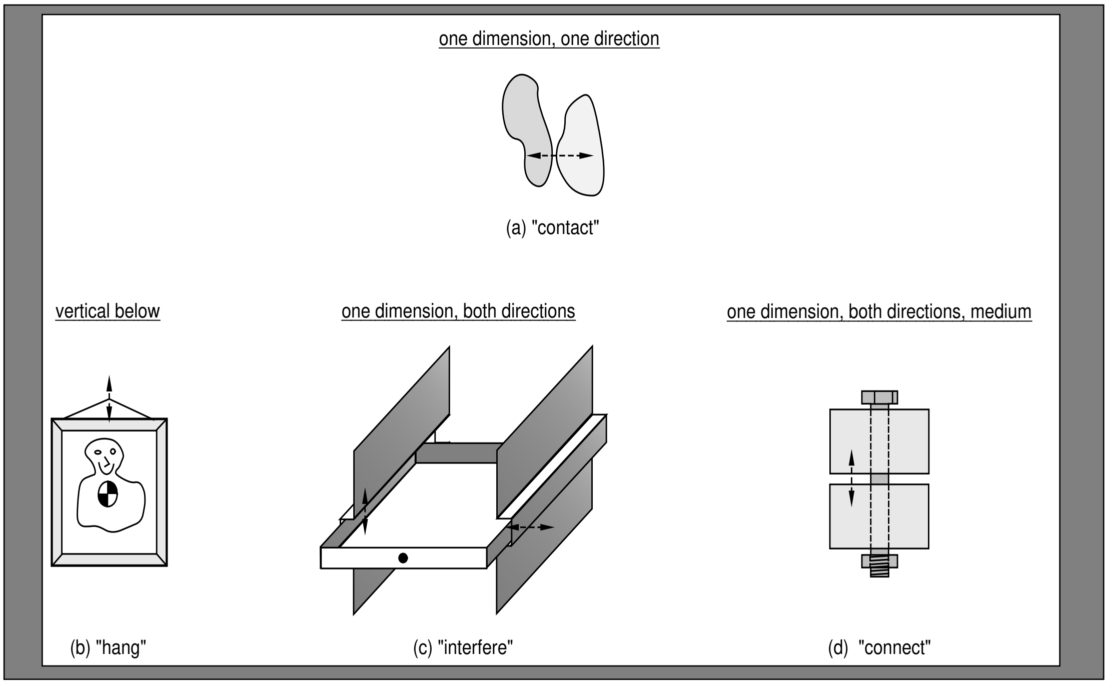


Figure 3.10 RESTRAIN process class specializations, (a) contact, (b) hang, (c) interfere, (d) connect.

Fig. 3.10 illustrates degrees of contact and their effect on object restraint. The arrows represent the dimension of contact between the two objects. Simple contact, shown in Fig. 3.10a, produces a restraint in a single dimension and direction. By specifying which dimension, number of directions, the relative placement of the two objects, and the presence of an instrumental object, such as a nut & bolt, three basic RESTRAIN class specializations: contact, interfere, and connect can be defined.

RESTRAIN-CONTACT: Motion for objects in contact is disabled in a single direction. The hanging picture illustrated in Fig. 3.10b is an example of RESTRAIN-CONTACT, called HANG, where the picture center of gravity is placed below the contact location.

RESTRAIN-INTERFERE: Contact between objects prevents motion along an entire dimension. One example of interference is an object sliding or rolling inside a channel where its motion is constrained to the channel by the channel walls. The desk drawer shown in Fig. 3.10c can only slide along the slot in which it fits; its motion is otherwise disabled by the slots on either side of it.

RESTRAIN-CONNECT: Objects which are connected lose mobility in multiple dimensions. A third object acts as an instrument, or medium, to fasten or join the objects (such as tacks, bolts, and clamps). The bolted blocks shown in Fig. 3.10d are held together by a nut and bolt, where the function of the nut and bolt is instrumental to the connection.

The hierarchical relationships between the generic RESTRAIN process, CONTACT, and the other RESTRAIN subclasses, INTERFERE, and CONNECT, are depicted in Fig.

3.11.

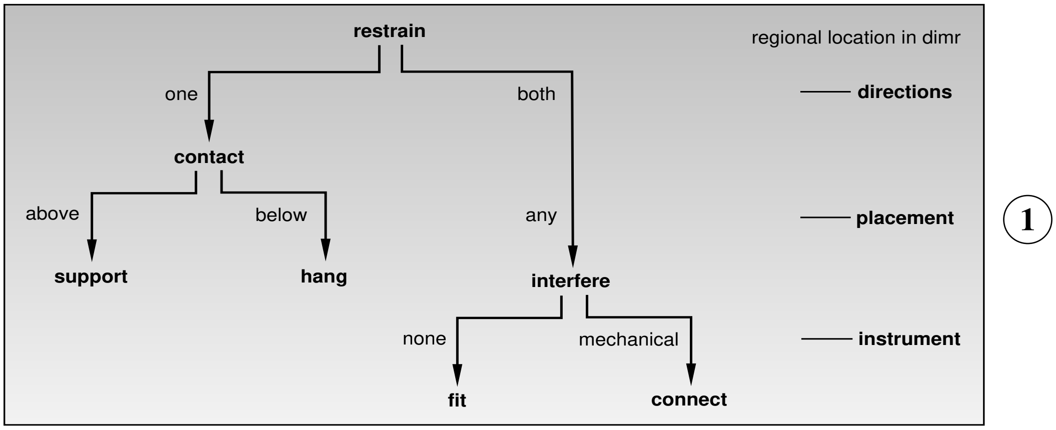


Figure 3.11 Inheritance hierarchy for RESTRAIN process, specialized by process role instantiations.

Specializing RESTRAIN processes enables representation of increasingly constrained motion. The process subclasses depicted in Fig. 3.11 are specialized according to how the behavioral process preconditions, specifically the statics-related preconditions, are instantiated. The process RESTRAIN-CONTACT can represent objects supported by or hanging from other objects, by specifying the dimension of contact to be vertical and by specifying the relative placement (1) of the two objects. These relationships are also shown in Table 3.3. In both HANG and SUPPORT, the process disables motion in the vertical dimension and the downward (NEG) direction. The supported object's center of gravity is placed above the restraining object, and the hanging object's center of gravity is placed below the restraining object.

Restrain Specialization	Restrain Class	Dimension	Placement	Instrument
SUPPORT	Contact	Vertical	Above	None
HANG	Contact	Vertical	Below	None
FIT	Interfere	Any	None	None
TIE	Connect	Any	None	Rope
BOLT	Connect	Any	None	Nut & Bolt

Table 3.3 Specializations of the RESTRAIN behavioral process primitive.

The process RESTRAIN-INTERFERE is used to represent objects which fit together. Two objects fit along a particular dimension when they contact in at least two colinear points. The result is restraint along the entire dimension of contact, which disables motion along the entire dimension of contact. The process RESTRAIN-CONNECT can represent objects which are tied or bolted together. In both INTERFERE and CONNECT, motion is disabled along the entire dimension and, depending on the object statics, along multiple dimensions. The instrumental object mediates the connection by providing the interference, rather than the restraining object. In RESTRAIN-SUPPORT and RESTRAIN-HANG, instrumental objects *can* be used; however, in RESTRAIN-CONNECT instrumental objects *must* be used. This is why no instrument is specified for the former constructs in Table 3.3. As an example of a required instrumental object, RESTRAIN-CONNECT is represented in frame notation below (Fig. 3.7) for the two bolted blocks depicted in Fig. 3.10d.

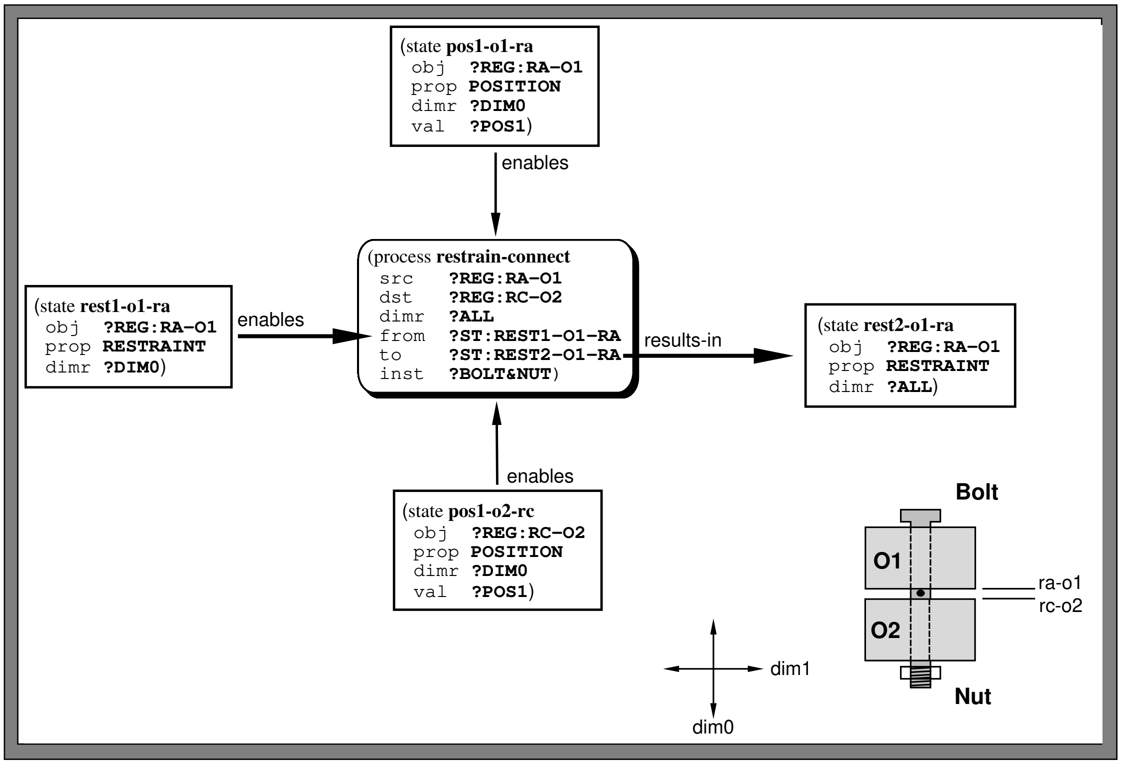


Figure 3.12 Graphical representation of CONNECT process specialization of the RESTRAIN process class.

In Fig. 3.7, the representation for RESTRAIN is shown instantiated for the subclass RESTRAIN-CONNECT**.** The states POS1-O1-RA and POS1-O2-RC represent the position of RA-O1 and RC-O2, in dimension DIM0 and value POS1. The bolt and nut provide additional contacts, and the overall result is full restraint in DIM0. In order for the bolt and nut to assist in restraining the blocks, independent RESTRAIN processes must be composable into generalized RESTRAIN processed. That is, RESTRAIN should be transitive.

RESTRAIN Process Combination

Processes can be combined to describe similar, albeit increasingly complex, behavior. RESTRAIN-INTERFERE and RESTRAIN-CONNECT restrain objects along entire dimensions. In some cases, such as a peg in a hole, interference results from multiple contacts between two objects. By combining the effect of multiple RESTRAIN processes, interference and connection involving more than two devices can be represented. The bolted blocks in Fig. 3.7 can only be represented with RESTRAIN-CONNECT with the help of a nut and bolt. This can be accomplished by combining two additional RESTRAIN-CONTACTs, in opposite directions, one for each block. This transitivity can also be applied to describing interference. Recall the nutcracker device presented on Page 86. One subgoal in using a nutcracker for cracking a pecan open is to restrain the pecan in the dimension in which the handles will be pressed together. Pecan restraint is achieved with a RESTRAIN-

INTERFERE-JAWS between JAW-L1 and the JAW-L2, both in the dimension of pressing, as illustrated in the lower diagram of Fig. 3.13.

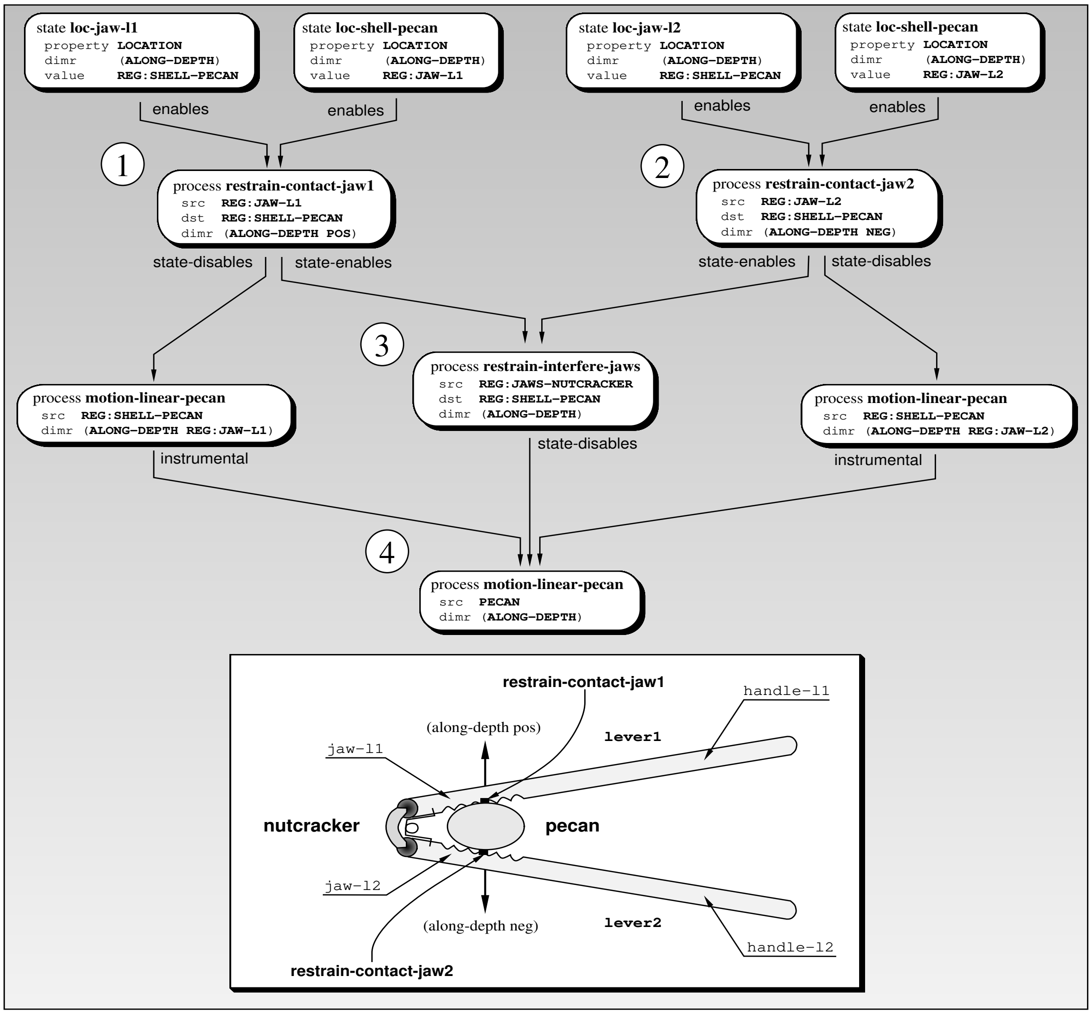


Figure 3.13 RESTRAIN-CONTACT and RESTRAIN-INTERFERE processes, instantiated for pecan nutcracking with a nutcracker.

The upper diagram in Fig. 3.13 represents the nutcracker jaw and pecan shell locations which enable RESTRAIN-CONTACT-JAW1 (1) and RESTRAIN-CONTACT-JAW2 (2). RESTRAIN-CONTACT-JAW1 results in a state which disables linear pecan motion in the (ALONG-DEPTH POS) dimension/direction. RESTRAIN-CONTACT-JAW2 results in a state which disables linear pecan motion in the (ALONG-DEPTH NEG) dimension/direction. The two processes combine to enable the RESTRAIN-INTERFERE-JAWS (3), whose resulting state disables linear pecan motion in the (ALONG-DEPTH) dimension (4).

3.2.4 TRANSFORM

The TRANSFORM process describes transmission of force between objects. Force magnification is a specialization of TRANSFORM which is used to represent mechanical advantage. Mechanical advantage is the principle by which mechanical devices function, so

TRANSFORM represents how mechanical devices work.

TRANSFORM involves two objects. One object is either in motion or is forced. This object comes into contact with a second object, and transmits force to the second object, as illustrated in Fig. 3.14. The process has two behavioral enablements: an applied force on the forced or moving object, and a restraint state resulting from a RESTRAIN-CONTACT between the two objects, both of which are defined in the same dimension/direction. When elastic objects (see item 1 in Fig. 3.14) come into contact (2), force is always transmitted, so the generic TRANSFORM process is TRANSFORM-TRANSMIT (T). An enabled TRANSFORM-TRANSMIT results in an applied force state on the unforced object in the same dimension/direction of, and having the same value as that of the forced object (3).

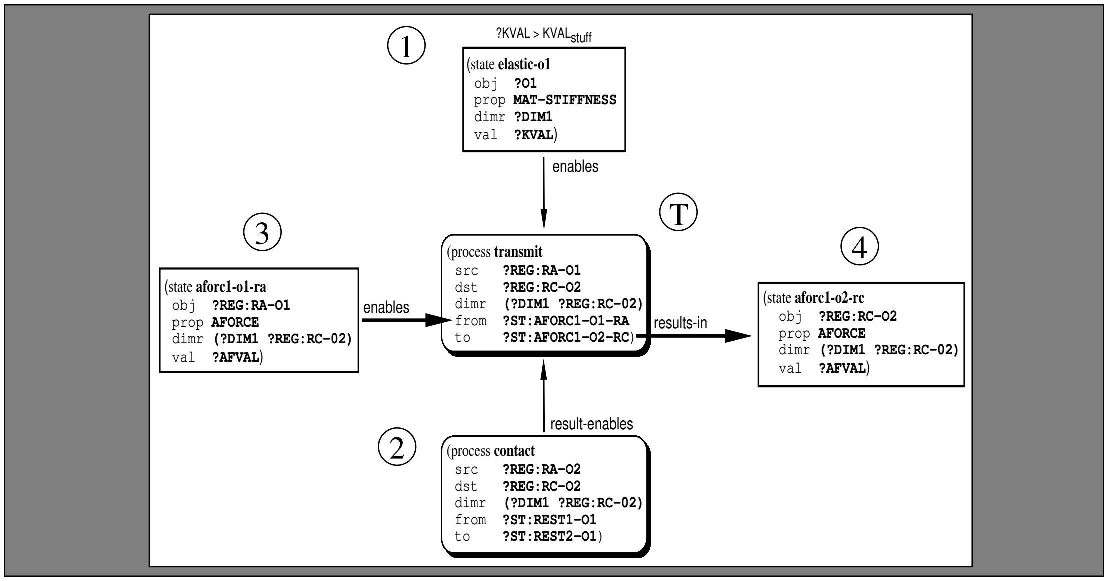


Figure 3.14 Frame representation for the TRANSFORM behavioral process primitive.

In Fig. 3.14, the state AFORCE-O1-RA (3) represents the process' perturbation behavioral enablement, applied at region RA on object O1. The dimension/direction of applied force is DIM1, and the associated value is AFVAL. The reaction to the applied force and contact between these objects is an applied force, AFORCE-O2-RC (4), at region RC on object O2. AFORCE-O2-RC has the same dimensions and value role bindings as AFORCE-O1-RA. In the figure, these applied force states are denoted with solid causal links to indicate that they are the states affected by the dynamic interaction.

The generic TRANSFORM process can be used to represent other force transformations by specializing the object statics of the transmitting object or the perturbation preconditions. For example, three transformation types are illustrated in Fig. 3.15. Force transmission (Fig. 3.15a) describes uniaxial force transfer from one object to another, and requires the two objects to be in contact in the same dimension and location as the applied force (an on-axis force). In the figure, force transmission is illustrated for both linear and rotational forces, as are the other transformations illustrated (force translation, Fig. 3.15b, and magnification, Fig. 3.15c).

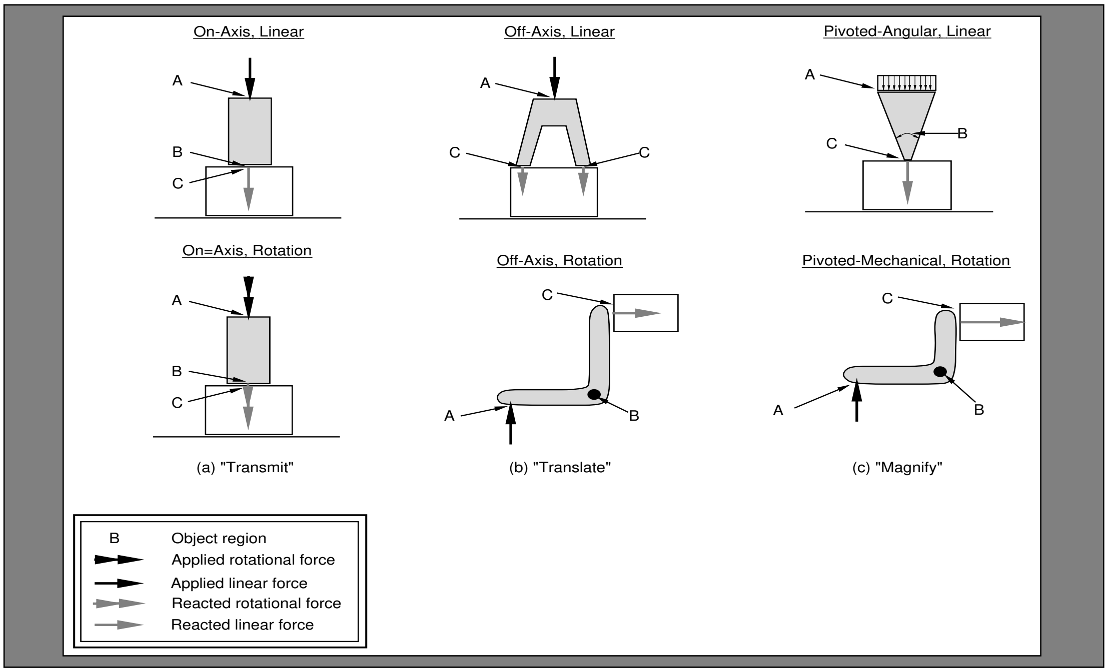


Figure 3.15 Illustrations for three TRANSFORM process specializations, (a) TRANSMIT, (b) TRANSLATE, and (c) MAGNIFY.

In Fig. 3.15, dark arrows represent applied forces (at A regions), while light arrows represents reacted forces (at C regions). Arrow length indicates the relative magnitude between the applied and reacted forces. For example, force transmission never changes dimension, direction, or magnitude, while magnification always changes magnitude.

TRANSFORM-TRANSMIT represents linear uniaxial force extension through transmission. The simplest example is a stick, where the point of contact is in line with the applied force, called on-axis forcing. The force dimension, direction, and magnitude remain the same, as depicted in Fig. 3.15a.

TRANSFORM-TRANSLATE represents transmission of force in a translational dimension other than that of the applied force. This is called off-axis forcing. Using a stick as a lateral probe, such as to knock a frisbee from a tree, is an example of linear off-axis translation. Linear translation also occurs when there are two or more contact locations, such as depicted in the upper portion of Fig. 3.15b. In such cases, the applied force magnitude is divided among the contact regions. Rotational force translation is similar to magnification, but the distances through which the applied force acts are equal.

TRANSFORM-MAGNIFY represents transmission of force from one region to another, whereby mechanical advantage is gained as a ratio of applied and reacted forces to the distance through which the applied and reacted forces travel, as shown in Fig. 3.15c. A simple example of rotational magnification is a crowbar, which has a handle where the force is applied, a fulcrum, and a reaction point where the crowbar comes into contact with the object to be pried. The crowbar rotates about the fulcrum, which remains stationary. The crowbar tip is located opposite the handle from the pivot, so it moves in the opposite direction. The distance between the tip and the fulcrum is less than the distance between the handle and the fulcrum, so the applied force is magnified.

Force magnification requires a pivot. There are three kinds of pivots: (1) a fulcrum pivot, which describes a simple contact in a dimension and allows rotation but not translation in the specified direction, (2) a fixed pivot which allows rotation in one or more dimensions but no translation, and (3) an offset pivot which is part of the object's shape definition. The angle on a wedge-shaped object, opposite the location of applied force, is called its *offset*. The upper example in Fig. 3.15c depicts a wedge object, and illustrates *translational magnification* using an offset pivot. The applied force is distributed over a surface, describing a pressure. The offset angle reduces the effective force reaction area, magnifying the pressure. This relationship is shown below (TR-1).

TR-1 AFVAL = A1/A2 * afval (3-1)

where AFVAL is the magnified force, afval is the applied force, A1 is the area where the force is applied, and A2 is the area where the force is reacted. The same relation can be calculated if the offset angle is known (TR-2), using the angle's tangent, or if the relative sizes of the wedge faces are known (TR-3):

TR-2 AFVAL = cotangent(offset) * afval (3-2)

TR-3 AFVAL = size(along-length)/size(along-width) * afval (3-3)

The lower example in Fig. 3.15c depicts a crank, which illustrates *rotational magnification* using a fixed pivot. The crank has a region which is stationary in the applied force dimension, but allows rotation about the applied force direction. Having its pivot located between the applied and reacted force regions results in a differential moment on the crank. Since the moments must sum to zero for static equilibrium (in Applied Mechanics), there is an effective magnification of the applied force at the contact region (C). The degree of magnification, or *mechanical advantage*, is governed by controlling the distance through which the applied force acts. Formally, mechanical advantage is defined as a trade-off between the force applied to a task and the distance (or area) through which the force is applied (TR-4):

TR-4 AFVAL = L1/L2 * afval (3-4)

where L1 and L2 represent the relative distances between the applied force location and the pivot location, and between the pivot location and the reacted force location.

Each of the transformation types can be represented with a TRANSFORM class. The hierarchical relationships between the generic TRANSFORM process, TRANSMIT, and the other TRANSFORM subclasses, TRANSLATE, and MAGNIFY, are depicted in Fig. 3.16. The TRANSFORM classes shown in Fig. 3.16 are distinguished by how the process preconditions are instantiated. An applied force which is coaxial with the contact location is called an *on-axis force*, and distinguishes pure force transmission from force translation (1). Fig. 3.15c (upper) showed the effect of object type (shape) on transmission, which is illustrated in Fig. 3.16 at (2). When the applied force is lateral with the contact location and the object CG, it is called an *off-axis force*. An off-axis force applied to an object without a pivot and a single contact location is called a nudge (as opposed to a prod, which is an axial transmission). The effect of adding multiple contact locations, illustrated in Fig. 3.15b (upper) is shown as a magnification in Fig. 3.16 at (3). This is actually a linear reduction since the applied force is divided among the number of contact locations.

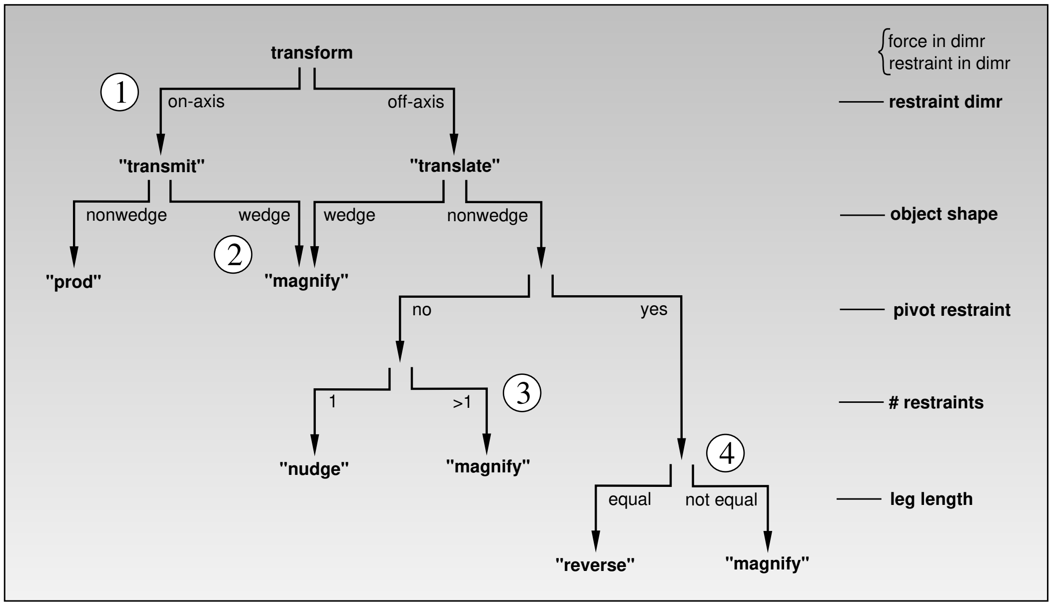


Figure 3.16 Inheritance hierarchy for the TRANSFORM process, specialized by process role instantiations.

Finally, the affect of adding a pivot, which was illustrated in Fig. 3.15b and c (lower), is shown in Fig. 3.16 at (4). The notion of leg length refers to the relative distances mentioned (as L1 and L2) in relations (TR-1) through (TR-4). When the leg lengths are equal, the effect of force transformation is to reverse the direction. When the lengths are unequal, then the result is magnification.

The relationship between applied and reacted forces for these TRANSFORM classes is shown in Table 3.4. The table is organized by the force dimension, direction, and magnitude.

Transform Class	Applied vs. Reacted Force		
	Dimension	Direction	Magnitude
TRANSMIT	Same	Same	Same
TRANSLATE	Same, Different	Same	Same
MAGNIFY	Same, Different	Same	Different

Table 3.4 Specializations to the TRANSFORM behavioral process primitive.

In the table, "same" means that the item is identical for both the applied and reacted forces. For example, the MAGNIFY class can represent forces which have the same, or different, dimensions, but the magnitudes are always be different.

The MAGNIFY subclass is used to represent all forms of force magnification described using equations (TR-1) through (TR-4). The differences between these relations depend on the object physical characteristics, so further discussion of mechanical advantage will be delayed until the next section. Since the distinction necessary to identify a MAGIFY transformation over a TRANSMIT or TRANSLATE depends only on the type of force and local restraint, MAGNIFY can be represented without knowing much about the objects in question. For example, the MAGNIFY process is shown instantiated for a rotational force magnification in Fig. 3.17. The pivot relationship for the object which will be magnifying the force, O1, must be in the dimension and direction of the applied force. In this example, this relationship cannot be represented with a fulcrum pivot, because translational motion is not constrained in DIM1 POS by a fulcrum. The process represented at (1) must be a fixed pivot.

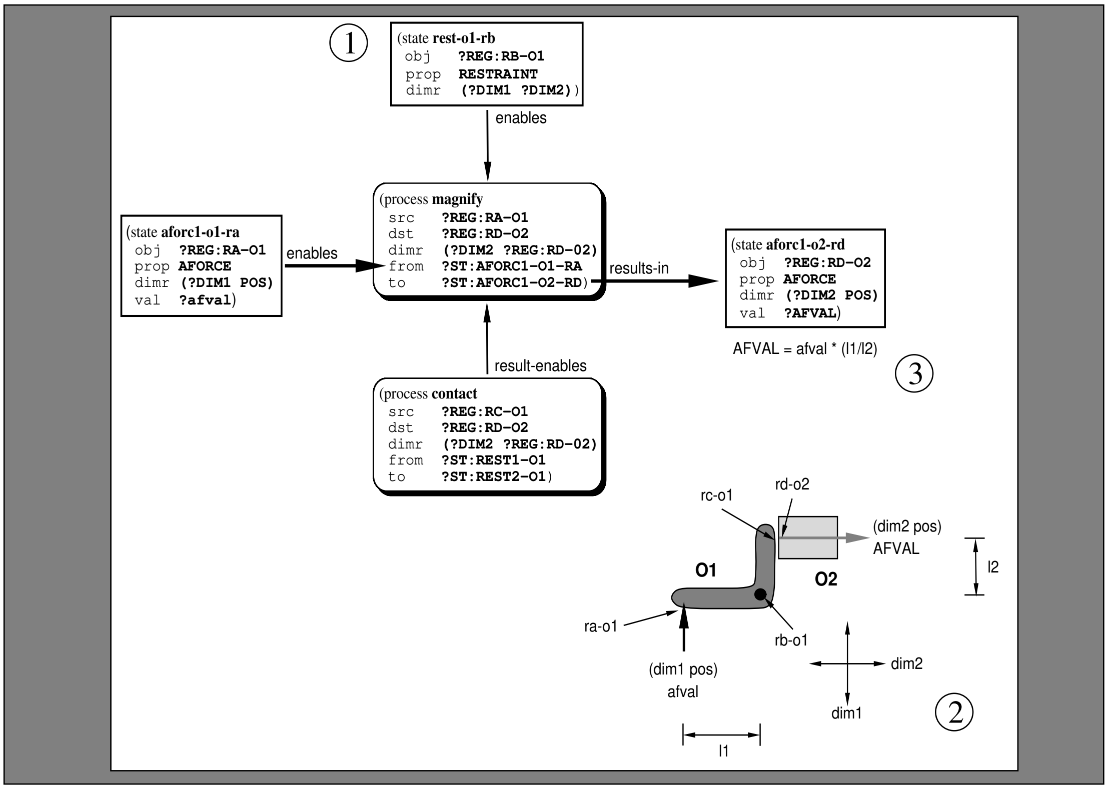


Figure 3.17 Graphical representation of MAGNIFY transform process.

The figure which accompanies this representation (2) depicts the relative distance between the force application location (RA-O1) and the pivot location (RB-O1) as l1, and the relative distance between RB-O1 and the reaction location (RC-O1) as l2. The force magnification produced by this process is represented by the aforce values afval and AFVAL. AFVAL is calculated, if l1 and l2 are known, by the (3-4) relationship illustrated under the reacted force state (3).

3.2.5 STORE

TRANSFORM describes interactions between objects where the object is considered rigid with respect to the magnitude of the applied force. That is, the TRANSFORM only describes the transfer of force from one object to another. The STORE process describes the effect of the new force on the reacted object when it is unable to move, such as compressing a spring between one’s fingers.

Each material has a limit, called an *elastic limit.*, or *yield strength*, depending on the units chosen to describe it. When an object is perturbed, and its motion is constrained, the applied force is absorbed internally. This absorption is called internal energy. The mechanical effect of energy absorption is a change in size or shape. This change is called deformation, or strain. Deformation is dependent on the amount of internal energy, so as the applied force increases, so does the deformation. The internal energy used to deform the object is released when the applied force is removed. As long as the applied force doesn't exceed the elastic limit, the object will return to its original size and shape when the force is removed. This is called *elastic deformation*. Internal energy is said to be *stored* in the object because it is the internal energy release which produces the reverse change in object size and shape. In FONM, the process of energy absorption and elastic deformation is distinguished from permanent, or *plastic*, deformation. The STORE process describes interactions which produce temporary deformation and energy absorption in objects.

The STORE process involves a single object and has two behavioral enablements: an applied force whose value doesn't exceed the object's elastic limit (shown at 1 in Fig. 3.18), and a restraint state on the object (2) which constrains its motion in the dimension/direction of the applied force. The enabled STORE process results in two states: an internal force which opposes the applied force (5), and a size in the applied force dimension (6). The change in internal force state is unique to the STORE process, while the change in size is also associated with DEFORM processes. The STORE process (S) organizes these interactions.

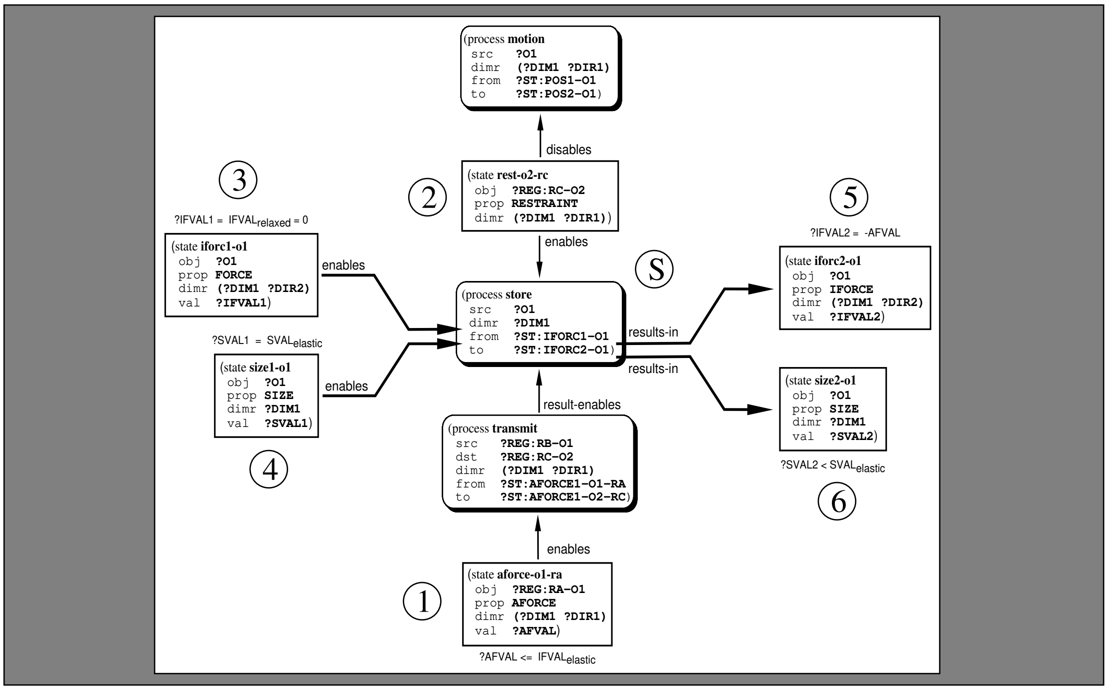


Figure 3.18 Frame representation for the STORE behavioral process primitive.

The IFORCE and SIZE states, labeled(3) and (4) in Fig. 3.18, represent the relaxed object, whose size is equivalent to the resting size for the object, and whose internal force is, subsequently, zero. The internal force resulting from STORE (5) has the same dimension as the applied force, but an opposite direction. The value, IFVAL, is identical to that of the applied force value, AFVAL.

STORE is used to represent how internal energy is stored and released in objects. It must be able to represent the effects of tension, compression, bending and torsion, which describe energy absorption in objects, as well as relaxation, which represents energy release in objects. The first four of these are illustrated in Fig. 3.19a-d. STORE is used to represent these forms of energy storage by specializing the characteristics of the applied force.

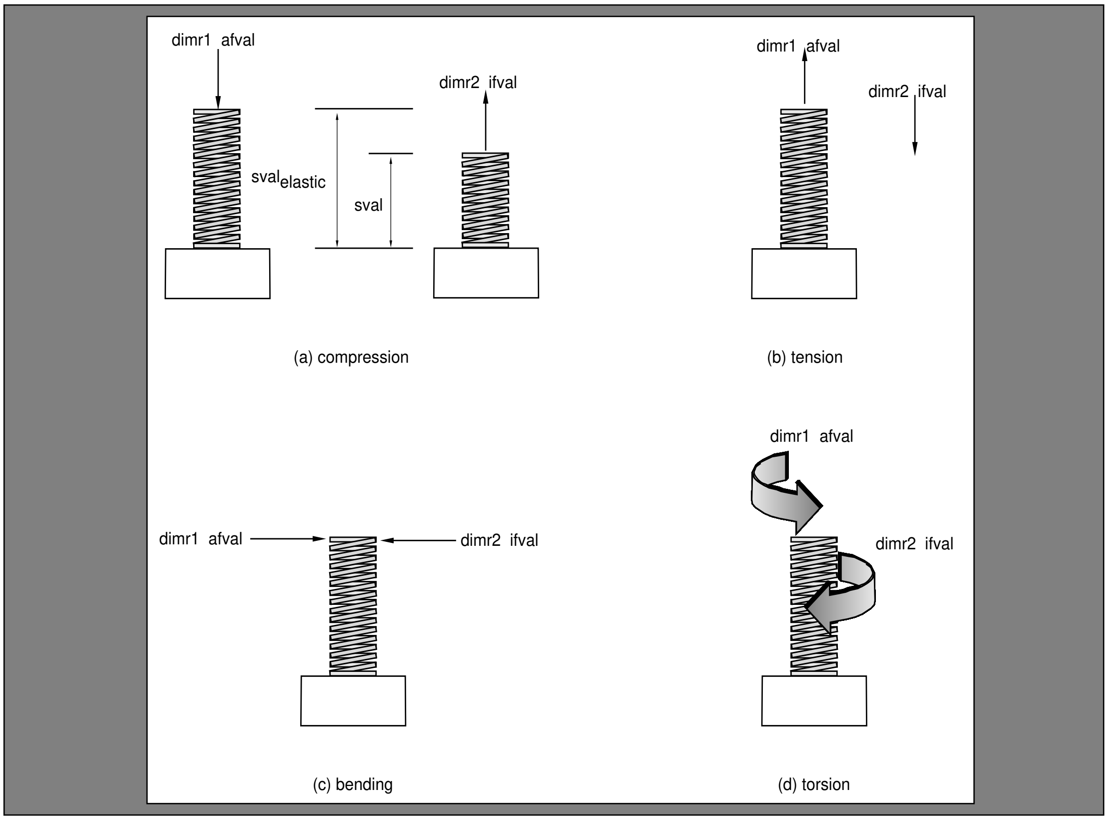


Figure 3.19 Specialization classes of the STORE behavioral process primitive, (a) compression, (b) tension, (c) bending, and (d) torsion.

The springs depicted in Fig. 3.19a, b are depicted with axial perturbations, and depict object compression and tension, respectively. The spring in Fig. 3.19c is perturbed laterally and depicts object bending. The spring in Fig. 3.19d is perturbed rotationally and depicts object torsion. DIMR1 represents the dimension and direction of the applied force, and AFVAL represents its magnitude. DIMR2 represents the dimension and direction of the internal force resulting from STORE, and IFVAL its magnitude. Each example in the figure describes the storage of internal force. When the internal force is initially nonzero and there is no applied force or its value is smaller than that of the internal force, then the object is said to be relaxing and describes the release of stored energy.

The $\text{SVAL}_{\text{ELASTIC}}$ and SVAL sizes shown in Fig. 3.19a represent the resting length or size of the object. When the object size in a particular dimension is equal to its resting length, then the STORE process doesn't predict any change in object internal energy. This means that the STORE process does not represent potential energy in objects.

The storage processes illustrated on the coil springs in Fig. 3.19a-d are represented as principal subclasses of the STORE process: STORE-COMPRESS, STORE-STRETCH, STORE-BEND, and STORE-TWIST. STORE-BEND and STORE-TWIST can both represent torsional compression or tension, depending on the applied force direction with respect to the object shape.

STORE-COMPRESS represents processes where axial force is applied toward the object CG, as with the coil spring in Fig. 3.19a. The process causes a size reduction with respect to the spring's resting length, and an internal energy, or force, directed away from the spring CG.

STORE-STRETCH represents processes where axial force is applied away from the object CG, as with the coil spring in Fig. 3.19b. The process causes a size increase with respect to the spring's resting length, and an internal energy, or force, directed toward the spring CG.

STORE-BEND represents processes where lateral force is applied away from the object longitudinal axis, as with the coil spring in Fig. 3.19c. The process causes a size increase with respect to the spring's resting length, and an internal energy, or force, directed toward the spring CG.

STORE-TWIST represents processes where rotational force is applied about the object longitudinal axis, as with the coil spring in Fig. 3.19d. The process causes a size increase or decrease with respect to the spring's resting length, and an internal energy, or force, in the opposite direction.

Compression and stretching describe energy *absorption* by an object. When the perturbation force is removed, the object is free to return to its original size by *releasing* the stored energy. Thus the concepts associated with the english terms "tension," "compression," "relaxation," and "torsion" are all represented with STORE. The hierarchy of STORE processes is shown in Fig. 3.20.

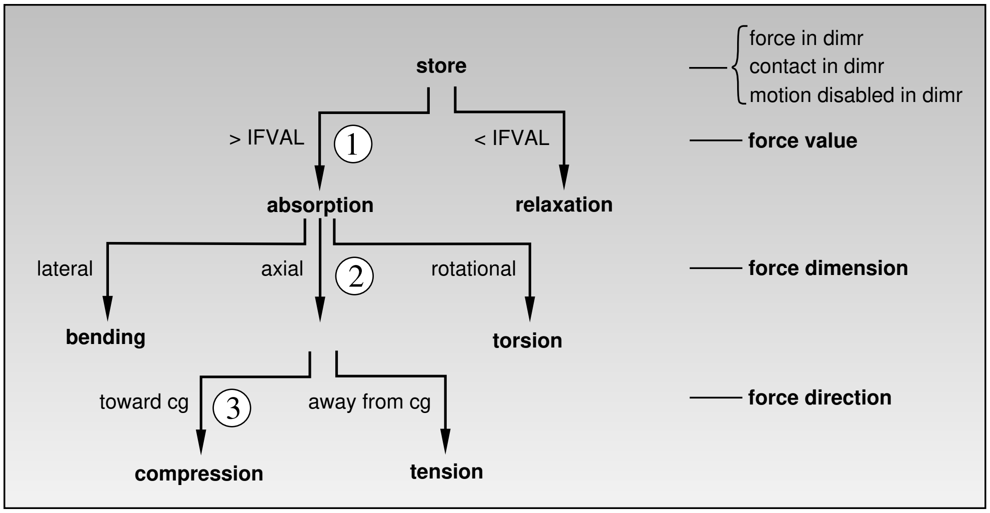


Figure 3.20 Inheritance hierarchy for STORE process, specialized by process role instantiations.

In this figure, the process subclasses are differentiated by how the process enablements are instantiated, specifically by the forcing type. For example, if the applied force is nonzero and greater than or equal to the value of IVAL (1), then energy is absorbed by the object. If the forced dimension is axial (2) and toward the object CG (3), then the object is in compression. These STORE classes are shown with their role instantiations in Fig. 3.5.

Store Class	Force Value	Force Dimension	Force Direction
STRETCH	AFVAL > IFVAL	Axial	(CG NEG)
COMPRESS	AFVAL > IFVAL	Axial	(CG POS)
TWIST	AFVAL > IFVAL	Rotational	Any
BEND	AFVAL > IFVAL	Lateral	Any
RELAX	AFVAL < IFVAL	Any	Any

Table 3.5 Specializations to the STORE process.

The frame representation for the STORE process is shown instantiated for the COMPRESS subclass in Fig. 3.21.

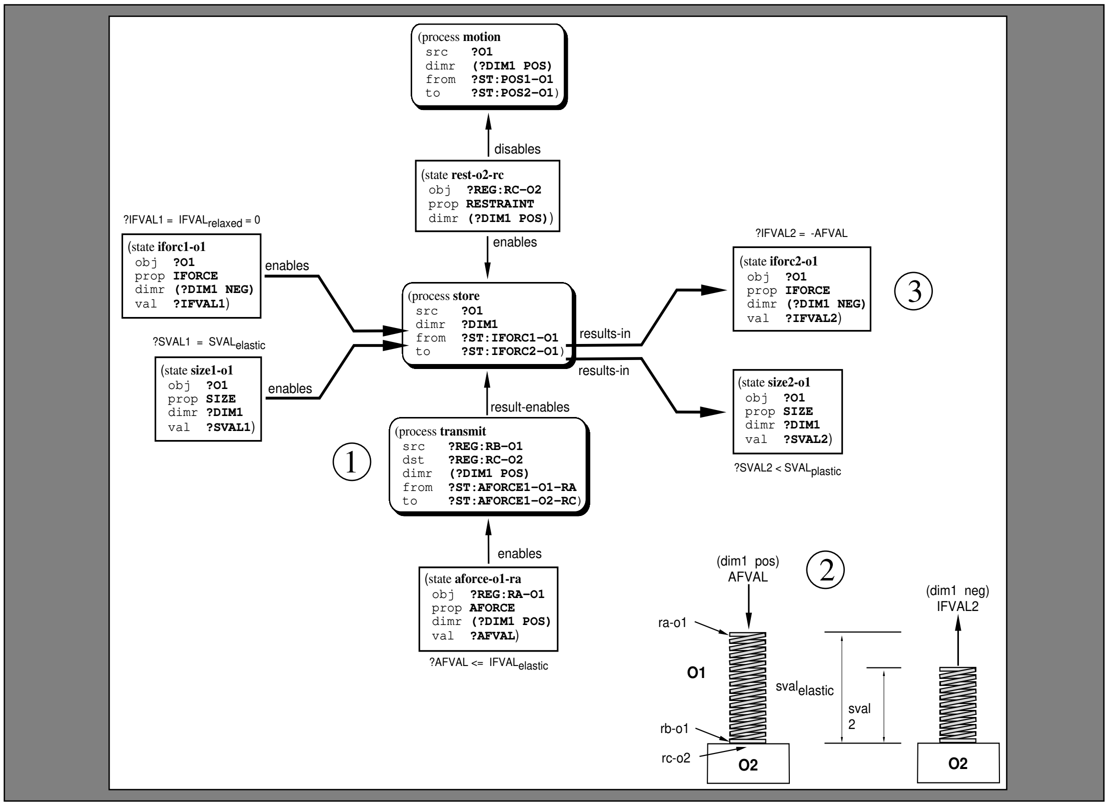


Figure 3.21 Graphical representation for the COMPRESS specialization of the STORE process class.

In the figure, AFORCE-O2-RC (not depicted in the figure) is the force that results from the TRANSMIT process (1) from O1 to the stationary object O2. AFORCE-O1-RA represents the perturbation force, as shown at (2), and is instantiated for dimension/direction DIMR1 and value AFVAL. In this case, the dimension is ALONG-LENGTH, and the direction is POS. The value AFVAL is less than or equal to the material elastic limit, $\text{IFVAL}_{\text{ELASTIC}}$. The states resulting from the enabled STORE-COMPRESS are IFORC2-O1 and SIZE2-

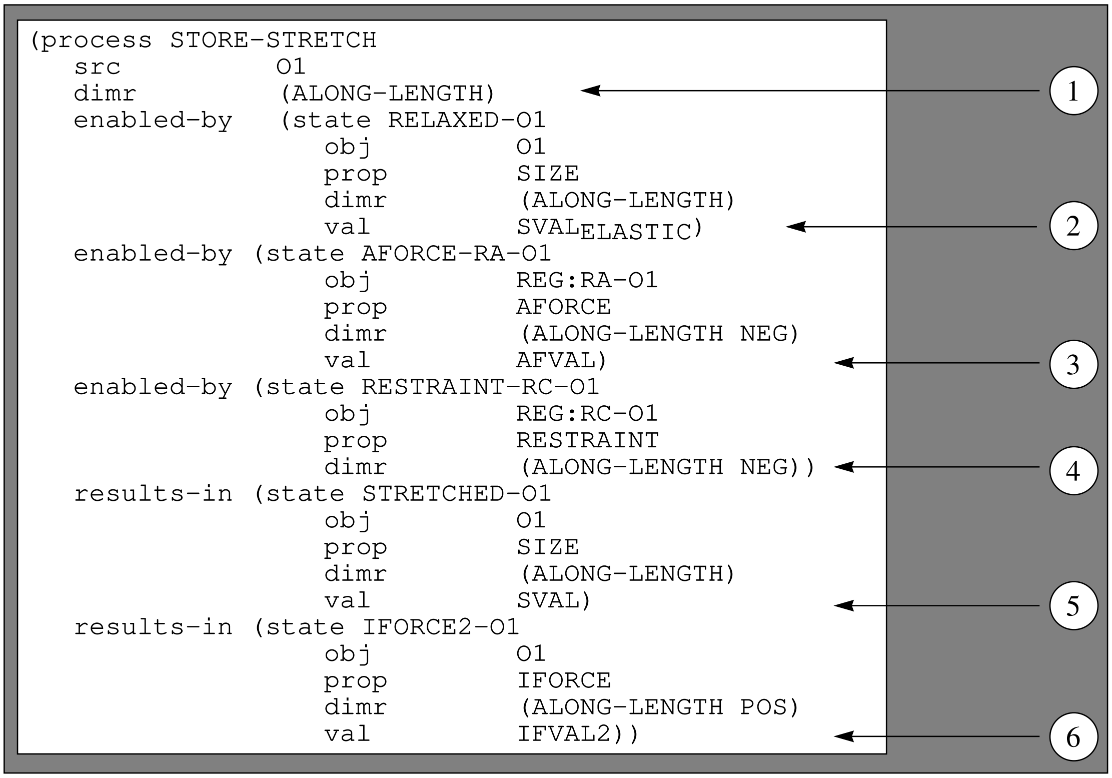
O1. IFORC2-O1 has dimension/direction DIMR2 which is opposite DIMR1 (ALONG-LENGTH NEG). IFORC2-O1 has a value IFVAL2 which is equal to AFVAL. SIZE2-O1 has the same dimension as DIMR1 and DIMR2, and a value SVAL2 which is less than the object's original size, or *resting length*, $\text{SVAL}_{\text{ELASTIC}}$.

The size state illustrated in Fig. 3.18 does not illustrate the relationship between the perturbation force value (AFVAL), an object’s resistance to deformation (or *spring constant*), and the reacted force (IFVAL). The relationship is shown below in SR-1:

SR-1: $\text{IFVAL} = K * X$ (3-5)

where K is the material spring constant and X is the object deformation. Further analysis of spring constants is ignored in MFRUP since the values of K and X are not easily measured, and the effect of applying force to an object is the same regardless of its value.

The relationship between applied force and resting length is represented for object stretching in Fig. 3.22, below. The naming conventions, regions, and dimensions in the figure are identical to those of Fig. 3.19a.



```
(process STORE-STRETCH
   src          O1
   dimr         (ALONG-LENGTH)
   enabled-by   (state RELAXED-O1
                   obj          O1
                   prop         SIZE
                   dimr         (ALONG-LENGTH)
                   val          SVALELASTIC)
   enabled-by (state AFORCE-RA-O1
                   obj          REG:RA-O1
                   prop         AFORCE
                   dimr         (ALONG-LENGTH NEG)
                   val          AFVAL)
   enabled-by (state RESTRAINT-RC-O1
                   obj          REG:RC-O1
                   prop         RESTRAINT
                   dimr         (ALONG-LENGTH NEG))
   results-in (state STRETCHED-O1
                   obj          O1
                   prop         SIZE
                   dimr         (ALONG-LENGTH)
                   val          SVAL)
   results-in (state IFORCE2-O1
                   obj          O1
                   prop         IFORCE
                   dimr         (ALONG-LENGTH POS)
                   val          IFVAL2))
```

Figure 3.22 The relationship between resting length, $\text{SVAL}_{\text{ELASTIC}}$, internal force, IFVAL, and length, SVAL.

Fig. 3.22 shows the representation for STORE-STRETCH, which is specified by the dimension and direction of applied force and restraint, (ALONG-LENGTH POS), shown at (3 and 4) which defines the process dimension (1). Not represented are the dependency relationships between STORE-STRETCH and STORE-COMPRESS, which are mutually exclusive, or the dependency relationships between the stretched, compressed, and relaxed bounding states. When an object is stretched its resting length is shorter than its actual length (5), resulting in an internal force (IFVAL2) greater than the prior internal force (6).

3.2.6 DEFORM

Objects undergoing deformations which exceed a material's elastic limit retain some deformation when the applied force is removed. This is called permanent, or plastic, deformation. The process DEFORM represents permanent object size and shape changes, such as those associated with bending, cutting, or breaking. DEFORM can be used represent the size or shape changes when a piece of clay is thrown onto the floor, extruded from a press, or cut with a knife.

The generic DEFORM process involves one or two objects and has two behavioral enablements: an applied force which exceeds the object's elastic limit (shown at label 1 in Fig. 3.23), and an internal force resulting from STORE (2), both in the same dimension/direction. The enabled DEFORM process (D) results in a new size state (3 5). The dimension of the size change is the same as the STORE process and the applied force. The value associated with the size change exceeds the material elastic strain limit, and will be greater than, or less than, the resting length in that dimension, depending on whether the perturbation force is toward, or away from the object center of gravity.

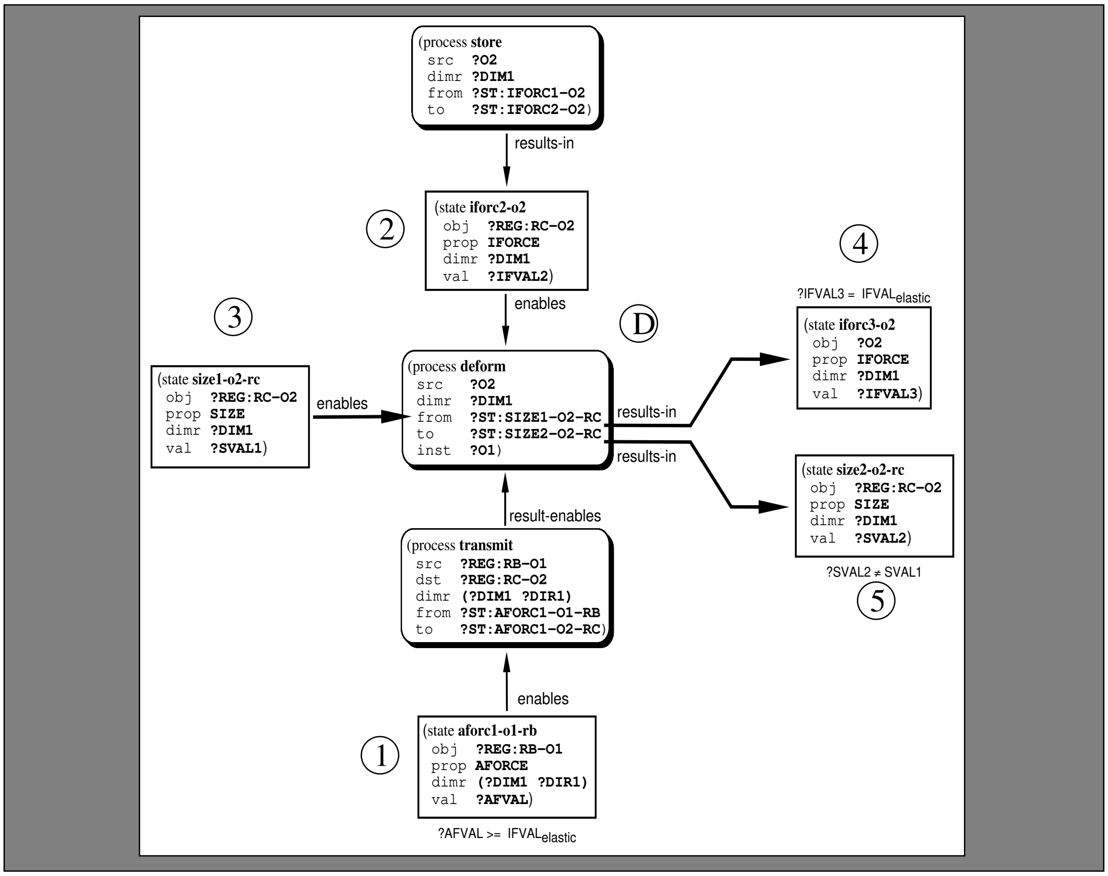


Figure 3.23 Graphical frame representation for the DEFORM behavioral process primitive.

In the figure, the resulting internal force has a value IFVAL3 = IFVAL$_{ELASTIC}$, which means that the internal force has reached its upper bound. When a cut is made, or an object is broken, material is removed from the original object and the internal force is released. The result is a new region, such as an indentation, whose size has the same dimension and value as the deformation. These examples of object deformation describe the kinds of behavior which DEFORM is used to represent, as illustrated in Fig. 3.24.

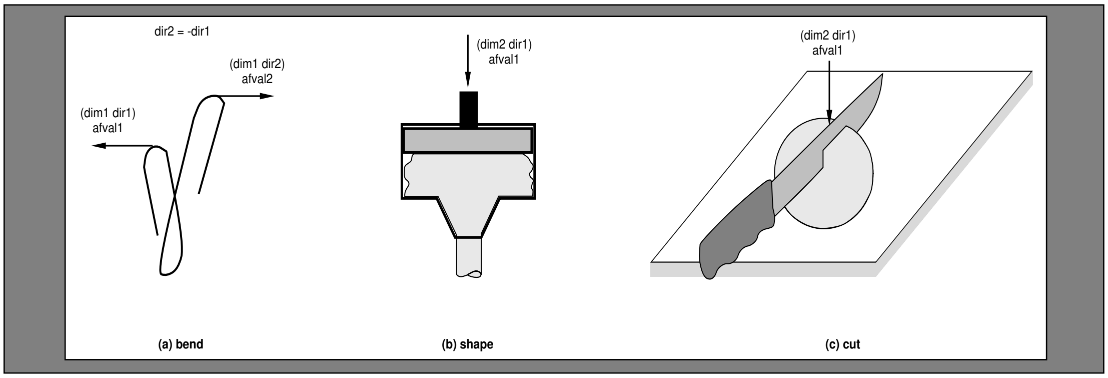


Figure 3.24 Three ways to mechanically deform an object: (a) bend, (b) shape, and (c) cut.

Three kinds of object deformation are shown in Fig. 3.24. The first (Fig. 3.24a) depicts the bending of a paperclip and illustrates a non-instrument deformation which results in a permanent shape change in DIM1, but no observable change in size or volume. Fig. 3.24b depicts the extrusion of some material from a manifold, which affects its shape and size but not its volume. Fig. 3.24c depicts the slicing of a tomato, which involves a knife as instrument and results in a cut in the tomato. The applied force in this instance exceeds the material's *breaking strength*, $\text{IVAL}_{\text{PLASTIC}}$. These three deformation types can be represented with the DEFORM process by prescribing the type and amount of deformation which the object undergoes: DEFORM-BEND, DEFORM-SHAPE, and DEFORM-CUT.

DEFORM-BEND represents object deformations in a single dimension which retain the object continuity. The applied force and restraint are in the same dimension and opposite directions. Bending results in shape changes but not noticeable size changes. For example, the paperclip length illustrated in Fig. 3.24a is increasing a small amount in dimr2 but appears to change a large amount in dimr1.

DEFORM-SHAPE represents object deformations in multiple dimensions which retain the object continuity, resulting in shape changes as well as size changes. The clay being extruded in Fig. 3.24b is undergoing shape and size changes in all dimensions, but is remaining a single piece of clay with the same volume.

DEFORM-CUT represents object deformations which affect object continuity, resulting in shape, volume, size, and, possibly, number changes. The tomato being sliced by the knife in Fig. 3.24c is changing size and shape. When the slice is completed, the tomato will be in two pieces.

The hierarchical relationships between these DEFORM classes are illustrated in Fig.

3.25 as specializations of the generic DEFORM process class.

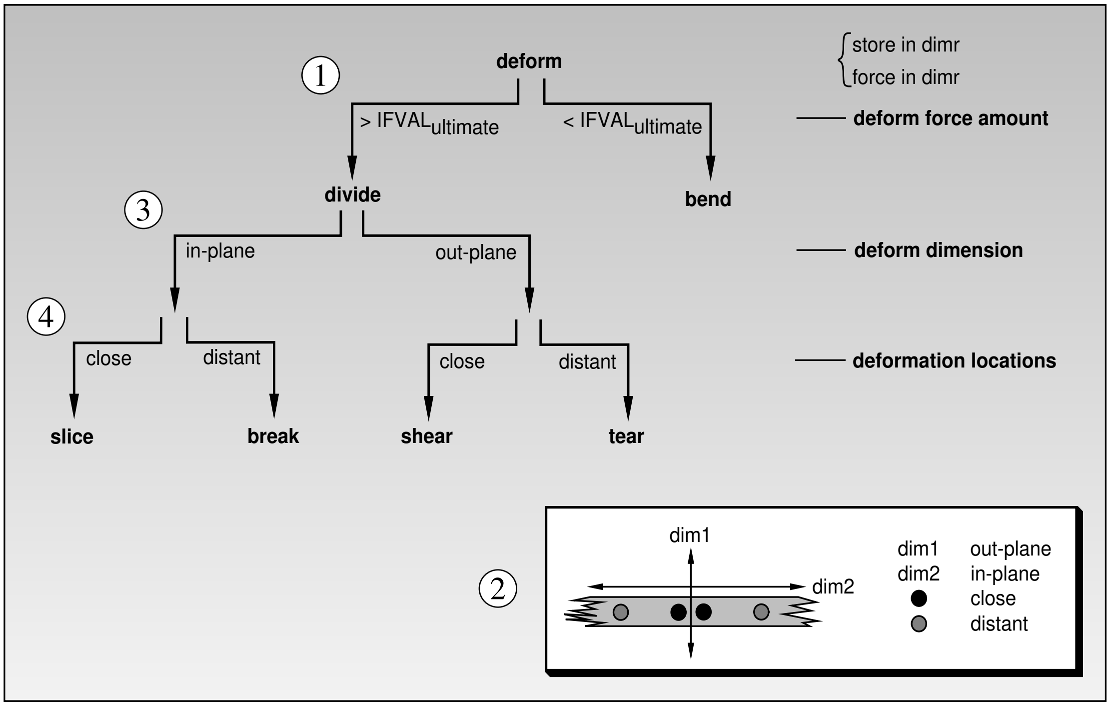
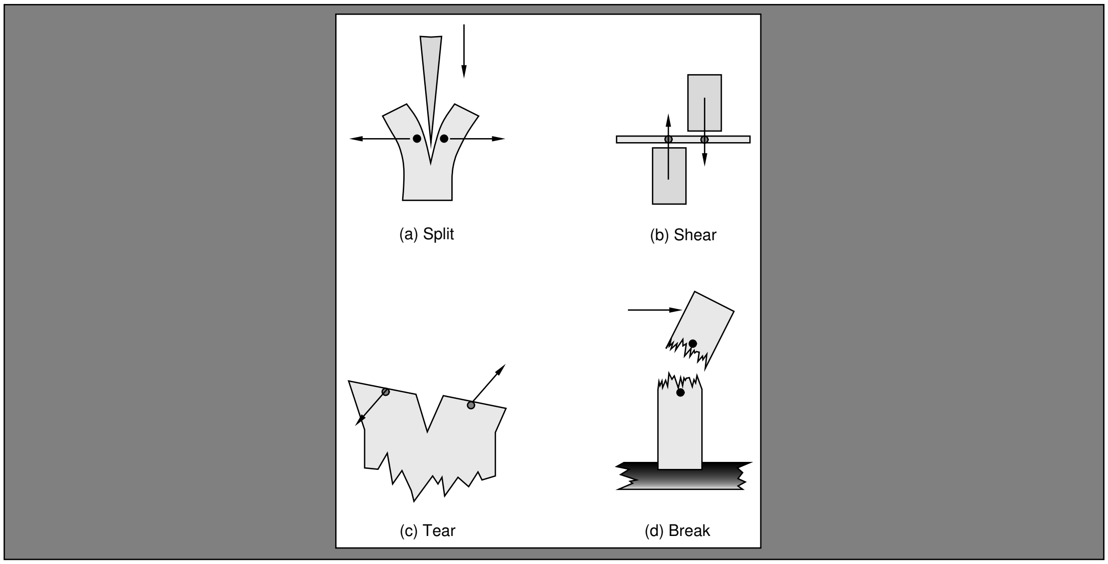


Figure 3.25 Inheritance hierarchy for DEFORM process, specialized by process role instantiations.

Processes in which the deformation exceeds the elastic limit (yield strength) but doesn't reach the plastic limit (breaking strength) are associated with bending. The paperclip bending illustrated in Fig. 3.24a instantiates DEFORM in this way. Other DEFORM instantiations are based on *dividing* material (shown at label 1 in Fig. 3.25). The legend (2) describes how divisions are defined. An *in-plane* division (3) is one in which the two material locations move away from each other in the material plane, such as in slicing and breaking. An *out-plane* division is one in which the two material locations move away from each other out of the material plane, such as in shearing and tearing. The division can be further defined by specifying how close the DEFORM locations are. Divisions where the perturbation force is physically close (4) to the deformed region, such as splitting and shearing, utilize instruments to force and restrain the object. When instruments are used, the resulting surface is *regular* or *smooth*. Divisions where the perturbation force is physically distant from the deformed region, such as tearing and breaking, result in *irregular* or *rough* surfaces. These forms of division are illustrated in Fig. 3.26a - d.


(a) Split

(b) Shear

(c) Tear

(d) Break

Figure 3.26 Graphical illustration for CUT specializations of the DEFORM process class.

The four division types illustrated in Fig. 3.26 can be represented as specializations on DEFORM-CUT by defining the deformation plane and the locations of restraint and applied force.

DEFORM-SPLIT represents a single wedge-shaped instrument forcing an object apart as in Fig. 3.26a. The translational magnification enabled by the wedge shape causes the force needed to exceed the plastic limit. The wedge faces then force the material [horizontally] away from the advancing wedge. SPLIT can be used to represent slicing and puncturing.

DEFORM-SHEAR represents the deformation caused by an instrument which impinges on another object from opposite directions of the same dimension. The perturbation force is directed toward the object CG, Fig. 3.26b.

DEFORM-TEAR is similar to shearing except that the object being deformed is not locally restrained, resulting in an uneven division. Also, the applied force is directed in opposite directions away from the object CG, Fig. 3.26c.

DEFORM-BREAK represents cuts which occur in lieu of bending. If the paperclip in Fig. 3.26a is replaced by one made of rigid aluminum, then instead of bending it would break, as in Fig. 3.26d.

Each of the DEFORM classes described above can be uniquely represented as a specialization of the process preconditions (behavioral and static), as shown in Table 3.5 below.

Deform Class	Instrument	Deform Dimension	Deform Locations	Length Change	Volume Change	Shape Change
BEND	No	Out-plane	Distant	No	No	Yes
SHAPE	Yes	N/A	N/A	No	Yes	Yes
SLICE	Yes	In-plane	Close	Yes	Yes	No
SHEAR	Yes	Out-plane	Close	Yes	Yes	No
TEAR	No	Out-plane	Distant	Yes	Yes	No
BREAK	No	In-plane	Distant	Yes	Yes	No

Table 3.6 Specializations to the DEFORM process.

The only characteristic missing from Table 3.5 is the applied force magnitude, which is between $IVAL_{ELASTIC}$ and $IVAL_{PLASTIC}$ for bending or shaping, and greater than $IVAL_{PLASTIC}$ for all division DEFORMs.

Using these definitions, the DEFORM representation for tomato slicing is presented in Fig. 3.27. The perturbation force is shown applied at RA-O1, which is a wedge-object in contact with object O2. Object O2 is in turn restrained from motion by O3. The force is magnified and transmitted to RC-O2 using TRANSFORM-MAGNIFY. The resulting force is applied at RC-O2 in dimension DIM1, direction POS, and value $AVAL > IVAL_{PLASTIC}$ (1). The force value and enabled STORE-COMPRESS enable the DEFORM process and identify the subclass as a DEFORM-CUT. The single instrument identifies the division as

DEFORM-SPLIT.

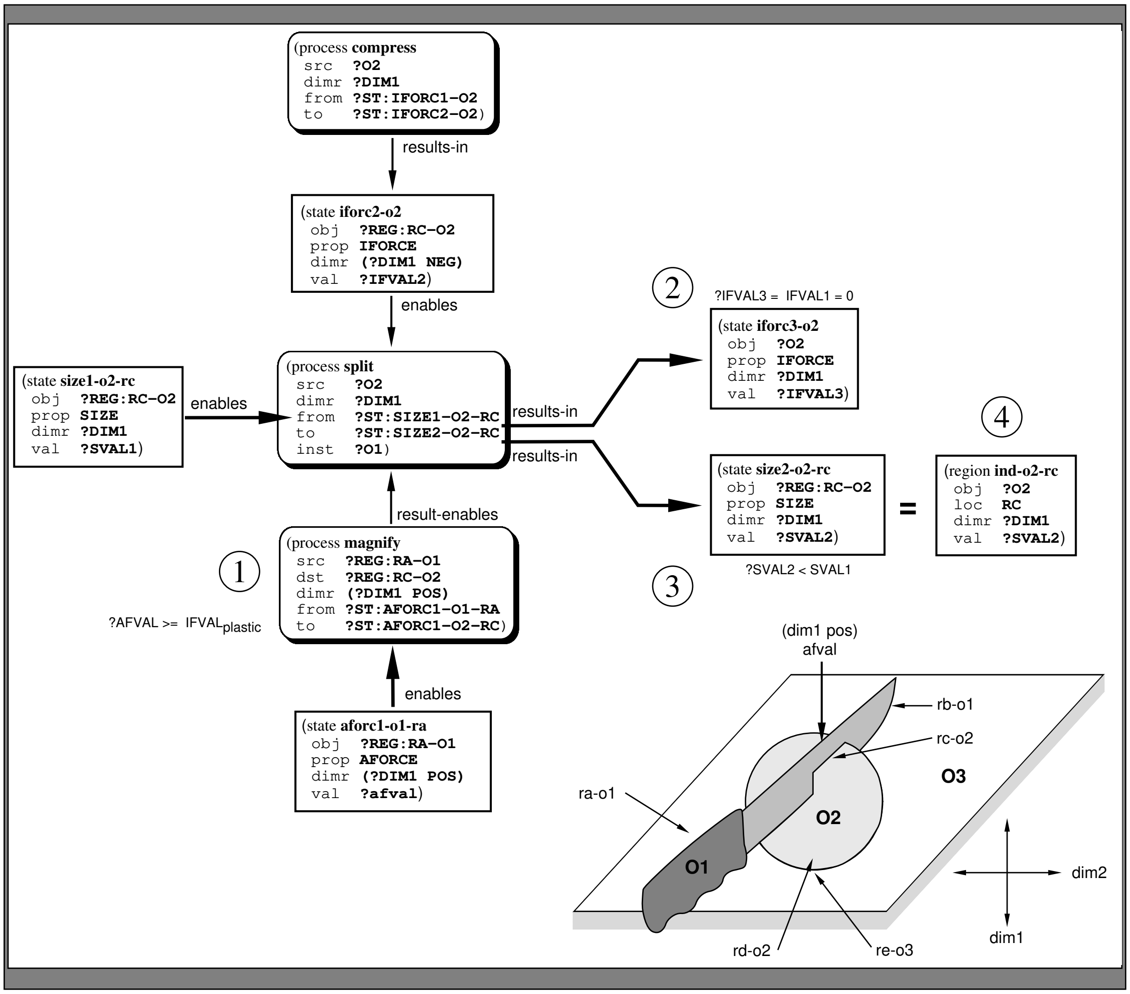


Figure 3.27 Graphical representation for the SLICE specialization of the DEFORM process class.

The states resulting from the enabled DEFORM-SPLIT are IFORC3-O2 (2) and SIZE2-O2 (3). The internal force state IFORC3-O2 has a magnitude of IFVAL3 = 0 because the split releases the force. SIZE2-O2 is defined in dimension DIMR1, and has a magnitude SVAL2 which exceeds the elastic strain limit $SVAL_{ELASTIC}$. The magnitude of SIZE2-O2 is less than the object's original size (SIZE1-O2) in dimension DIMR2. The change in size is attributed to the creation of a region whose size it reflects. In this example, the region is shown at (4).

It should be noted that the process illustrated represents the division of two locations on O1 as a single iteration in the slicing of a tomato and not the entire process, which constitutes a SLICE-APART function. In successive iterations, different locations are separated in DIM1, along DIM2, eventually resulting in separation of surface locations.

Complex cutting processes, such as chopping, clipping, sawing and mowing, can be represented by combining DEFORM specializations in the same way as interference was defined as the combination of RESTRAIN specializations. For example, sawing is a combination of slicing and breaking, where a wedge is forced into an object at an oblique angle and snaps a portion of the material off. Since there is a row of teeth which perform this process in unison, a line is sawed in the material. In FONM, the saw's edge is represented as a tooth region, so the DEFORM-CHIP process resulting from the combination of DEFORM-SPLIT and DEFORM-BREAK, represents the behavior.

3.3 Device Behavior Sequences

Combinations of like processes can be used to describe process specializations, as described with RESTRAIN-CONTACT and RESTRAIN-INTERFERE, and illustrated in Fig. 3.13. Device behavior is generally more complex than one process or the combination of instances from one process class. General device behavior can be a complex interaction of different process types, each with its own enabling conditions. Complex object behavior can be represented by combining BPPs in causal sequences. Consider the behavioral sequence associated with cracking pecans open with a nutcracker, illustrated in Fig. 3.28.

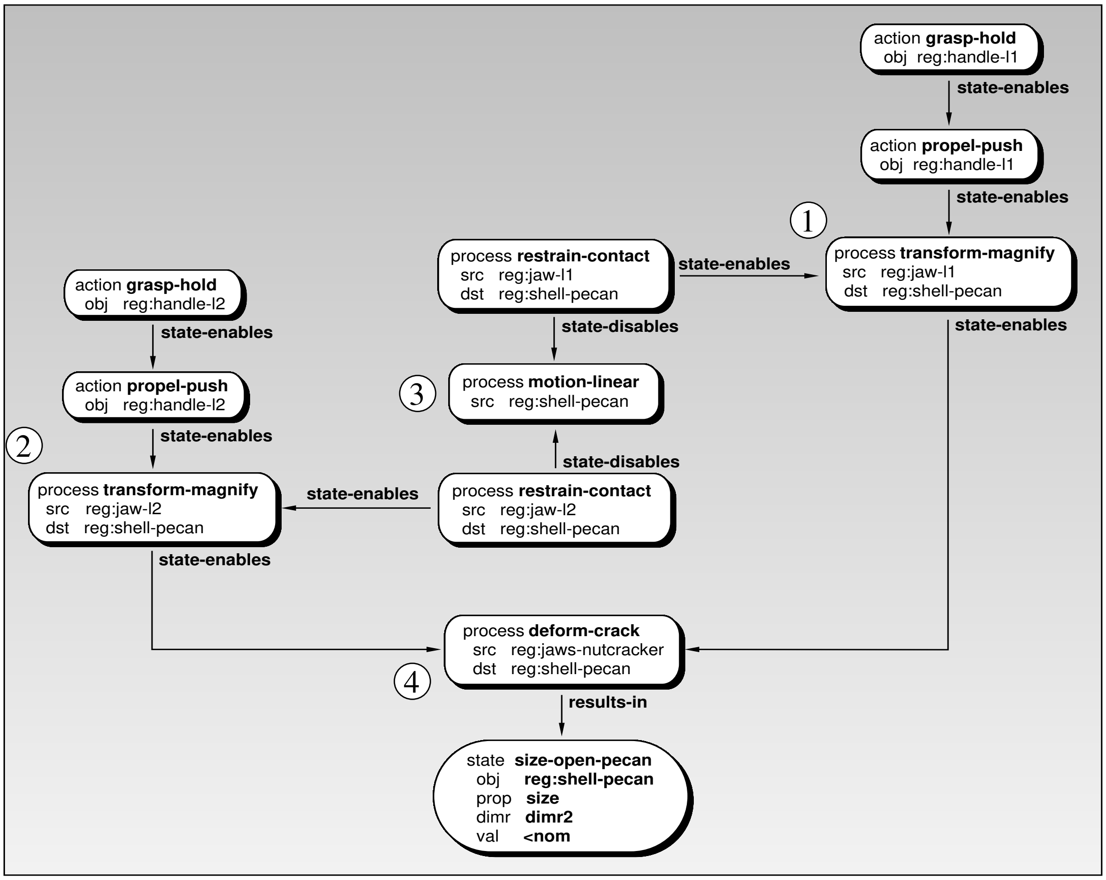


Figure 3.28 BPP sequences for representing nut cracking with a nutcracker

Fig. 3.28 illustrates the actions and processes which are involved in opening a pecan. Only the frame representation of each action or process is depicted, with the assumption that all action or process enablements are met. For example, the applied force states which enable TRANSFORM-MAGNIFY (at labels 1 and 2) are not depicted, but the relation between the PROPEL action, and the process TRANSFORM-MAGNIFY is labeled STATE-ENABLES. In FONM, a PROPEL action on an object causes an applied force on the object. The states are replaced with the STATE-ENABLES link, which is a notational shorthand for X results-in STATE, STATE enables Y, and is used to reduce the size and visual complexity of the illustration. Similarly, the position states which enable the two RESTRAIN-CONTACT processes, and the restraint states resulting from them have also been omitted. The storage of elastic energy in the nutcracker spring (STORE-COMPRESS) is not illustrated, and no dimensions or values are shown on any of the frames, except for the resulting pecan size state.

According to Fig. 3.28, the nutcracker behaves as follows. Holding and pressing the nutcracker handles are actions which produce an applied force state along the length of their respective handles (HANDLE-L1 and HANDLE-L2). The sequence illustrated assumes that a pecan is positioned between the nutcracker jaws (JAW-L1 and JAW-L2), which enables the two RESTRAIN-CONTACT processes. The pecan is thus constrained from linear motion in both directions, as represented in Fig. 3.13. This motion disablement has been collapsed to the single process MOTION-LINEAR (3) in Fig. 3.28. The applied force on the handles, and the restraint states on the pecan jointly enable force transmission to the pecan shell at the JAW-L1 and JAW-L2 regions. Because the nutcracker handles are pivoted at one end (not shown in this figure), a TRANSFORM-MAGNIFY is enabled, and the force transmitted to the pecan shell is also magnified. If the magnified force exceeds the pecan shell elastic limit, then a DEFORM-CRACK (4) will be enabled. The state caused by DEFORM is a change in size. This state is called the pecan open state, SIZE-OPEN-PECAN.

The BPP sequences which are associated with device function can be collapsed into a single frame by considering only the information required to enable and initiate the function and the states which result from successful function application. One such example is shown in Fig. 3.29, which illustrates two screwdriver functions, *PRY*-OPEN and *POUND*-CLOSED. These functions illustrate the difference in representational granularity between device behavior and device function. The illustration at the bottom of Fig. 3.29 depicts a screwdriver being used to pry open a paint can. Fig. 3.29b represents PRY-OPEN-PCAN as a (low-level) sequence of behavioral processes, and depicts all the processes necessary to understand how the can is opened (similar to Fig. 3.28).

The function PRY-OPEN-PCAN is initiated by holding and pushing the screwdriver handle (region HANDLE-SD) toward the paint can (object PCAN). These actions produce an applied force on the screwdriver. The screwdriver tip (region TIP-SD) is shown in contact (RESTRAIN-CONTACT) with the lip of the can (region LIP-PCAN-LID). The force and restraint states enable transmission (TRANSFORM-TRANSMIT) of the applied force from the tip to the can lid (TIP-SD to LIP-PCAN-LID). The resulting force, at LIP-PCAN-LID, is magnified (TRANSFORM-MAGNIFY) because of the second contact (RESTRAIN-CONTACT) between the screwdriver shaft and the can's lip (BODY-SD and LIP-PCAN-VESSEL) acts as a fulcrum (screwdriver rotates about the pivot, not shown). The force state at LIP-PCAN-LID can, depending on its value, enable the lid (PCAN-LID) to move upward (MOVE-LINEAR). If motion is enabled, then contact (RESTRAIN-CONTACT) between can and lid (PCAN-VESSEL and PCAN-LID) is disabled. The PCAN-LID location resulting from the disabled interference process (between PCAN-VESSEL and PCAN-LID) is the PCAN restraint state, REST-OPEN-PCAN. The overall PRY-OPEN-PCAN function can thus described as a combination of six BPP process specializations: CONTACT, TRANSMIT, ROTATE, MAGNIFY, LINEAR, and INTERFERE.

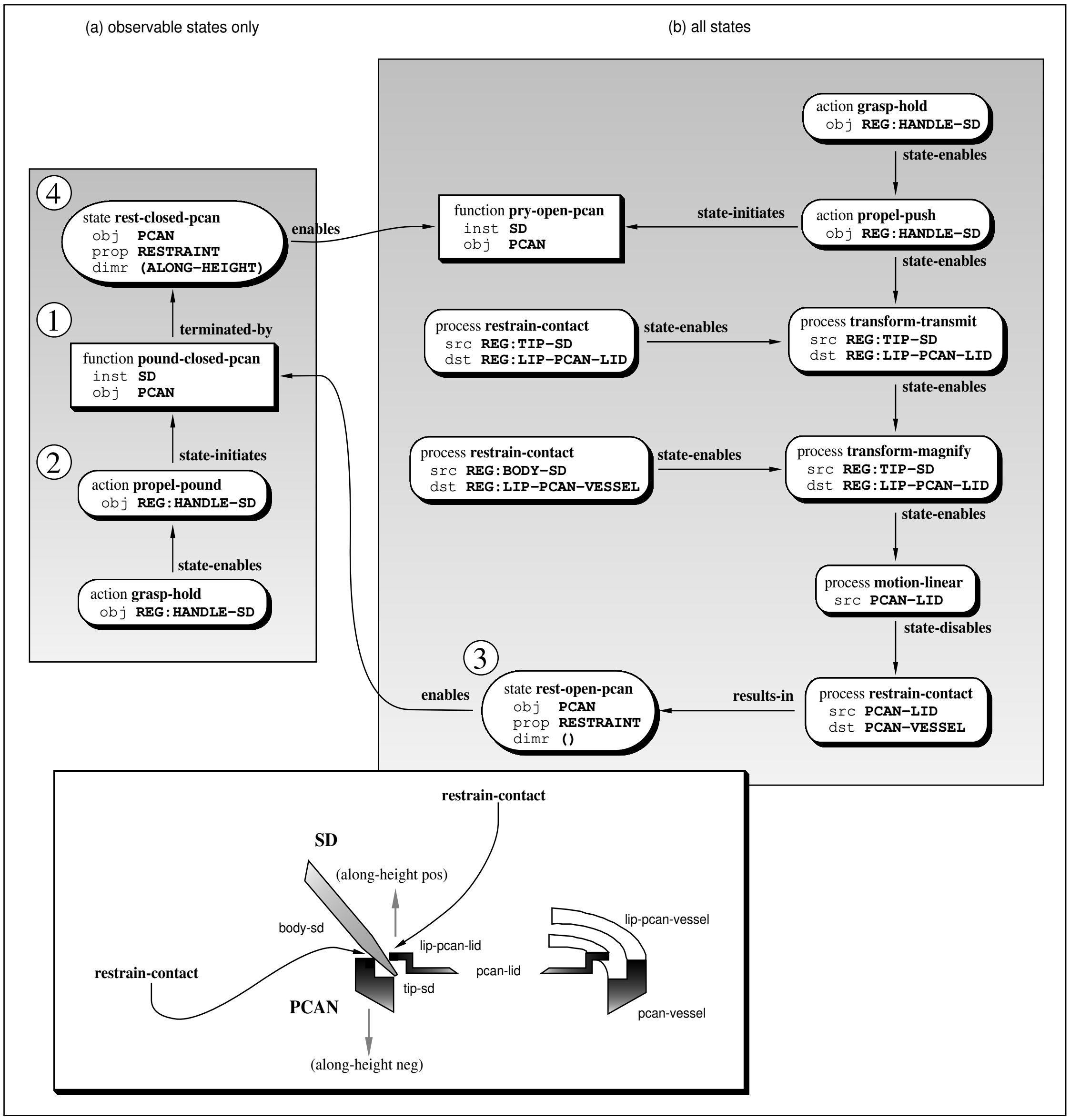


Figure 3.29 Behavioral sequences for representing the screwdriver functions associated with (a) pounding, and (b) prying a paint can lid.

Fig. 3.29a represents a second screwdriver function, that of pounding a paint can lid onto a paint can. The function POUND-CLOSED-PCAN (at label 1) illustrates a frame description for an entire device. POUND-CLOSED-PCAN is initiated by grabbing (GRASP) the screwdriver handle and pushing (PROPEL) the end of the handle (region END-SD) down (at 2) on top of the paint can lid (region TOP-SURFACE-PCAN-LID). POUND-CLOSED-PCAN is enabled by the can being open (state RESTRAINT-OPEN-PCAN, at 3), and by the placement (not shown) of the lid (PCAN-LID) above/onto the can (PCAN-VESSEL). POUND-CLOSED-PCAN is terminated by the state REST-CLOSED-PCAN (at 4) between the lid and the can.

The behavioral sequences shown in Fig. 3.28 and Fig. 3.29 describe general patterns (functions) which can also be instantiated by other devices. For example, the function CRACK-OPEN using a nutcracker, shown in Fig. 3.28, can also be instantiated by a crank can opener as long as the can opener statics support the function, and as long as the device regions are properly matched. Similarly, a hammer could replace the screwdriver in the POUND-CLOSED-PCAN function of Fig. 3.29a.

The BPP sequences fleshed out in Fig. 3.28 and Fig. 3.29b, and the function frame presented in Fig. 3.29a can be described for any device, however, these representations place more emphasis on how a device behaves and less emphasis on a device's components and the roles they play in how the device behaves. In order to see why a device is constructed a particular way, its overall behavior should be represented in terms of the behavior of its components. Since a component may be used in other devices, recognizing its presence in a device enables its behavior to be inferred and applied as necessary. For example, consider the simple screwdriver shown in Fig. 3.30.

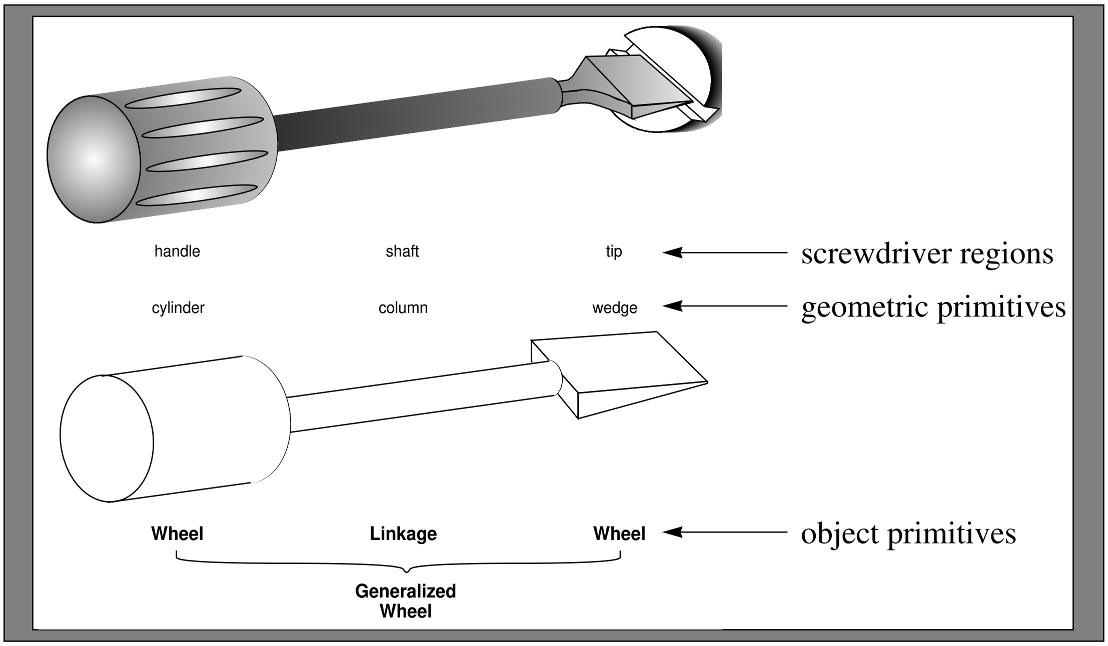


Figure 3.30 Illustration of a screwdriver and its idealized representation for driving screws.

The upper portion of Fig. 3.30 illustrates a screwdriver being used to drive a screw. The screwdriver handle, shaft, and tip components are labeled in the figure. The lower portion of the figure depicts the FONM idealization of the screwdriver as a simple device comprised of three geometric primitives: (1) a cylinder, (2) a column, and (3) a wedge. When the screwdriver handle is turned, as when driving a screw, a rotational force is applied to the handle. Because the handle and shaft are rigidly connected (see Fig. 2.63, Page 84), the rotational force is transmitted radially to the shaft, and, likewise, to the tip. The BPP sequence representations for the screwdriver components, for the driving function, are illustrated in Fig. 3.31-Fig. 3.33.

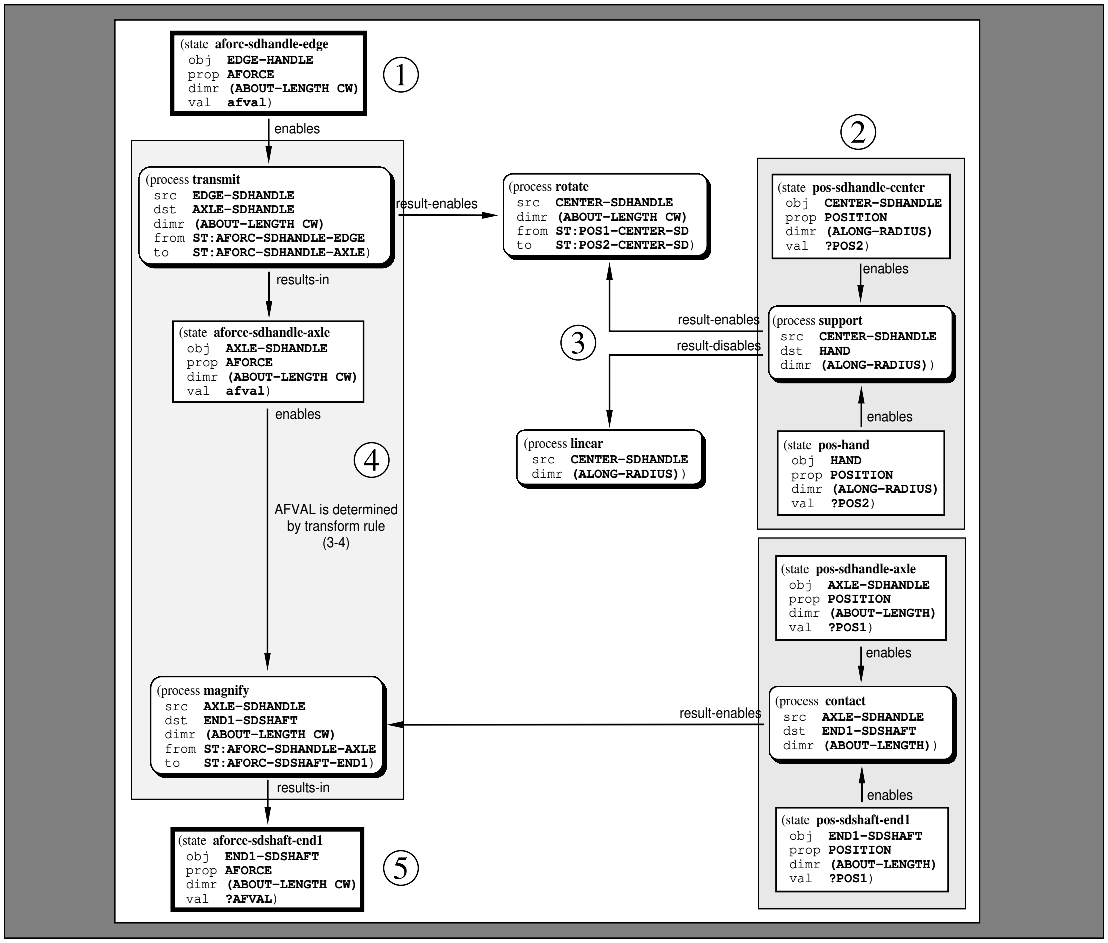


Figure 3.31 Frame representation for the screwdriver handle dynamics when the applied force is torsional.

Fig. 3.31 illustrates the effect of turning a screwdriver handle. The applied force caused by turning the handles is shown at (1). The hand holding the screwdriver enables a SUPPORT process (2), which is illustrated without the ABOVE placement between the screwdriver and the hand. The SUPPORT process provides the pivot which enables MOVE-ROTATE, and the restraint necessary to disable MOVE-LINEAR (3). The shaded box at (4) shows how the interactions between the handle and the shaft are combined into a BPP sequence describing screwdriver handle dynamics given the applied force and restraints. For this example, the handle rotates and the applied force is transmitted and magnified between the outer surface of the handle and the location where the handle and shaft are connected. The value of the force magnitude (AFVAL) resulting from this interaction is determined using transform relation (3.4). A similar set of processes is presented for both the screwdriver shaft and tip.

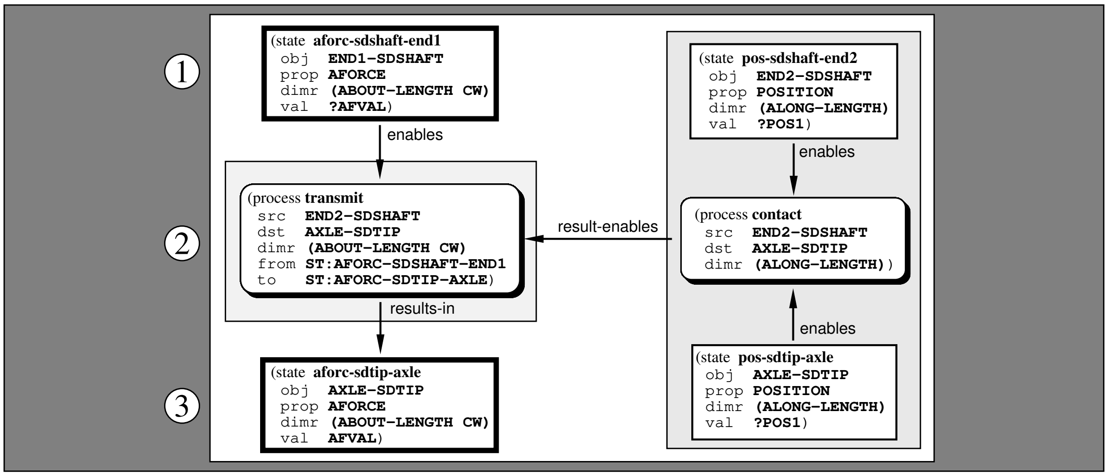


Figure 3.32 Frame representation for the screwdriver shaft dynamics when the applied force is torsional.

Fig. 3.32 illustrates the role the screwdriver shaft plays in the driving of screws. The shaft takes the force magnified by the handle and transmits it to the screwdriver tip. The state labeled (1) is identical to the state labeled (5) in Fig. 3.31. The shaded box (2) represents the BPP sequence associated with the screwdriver shaft. The single process is indicative of a very simple behavior, since the torsional force is only being relocated some distance away from where it was originally applied. Fig. 3.33 shows the role the screwdriver tip plays in the driving of screws. The BPP sequence begins (1) with the state labeled (3) in Fig. 3.32. The interactions shown are identical to those shown in Fig. 3.31, except that instead of a torsional force being applied at an outer radius and being transmitted to an inner radius, the reverse is occurring. The support relation still enables rotation, and the contact between the tip and the screw enables force transmission. The shaded box (4) represents the screwdriver tip dynamics, but the pattern is virtually identical to that for the screwdriver handle.

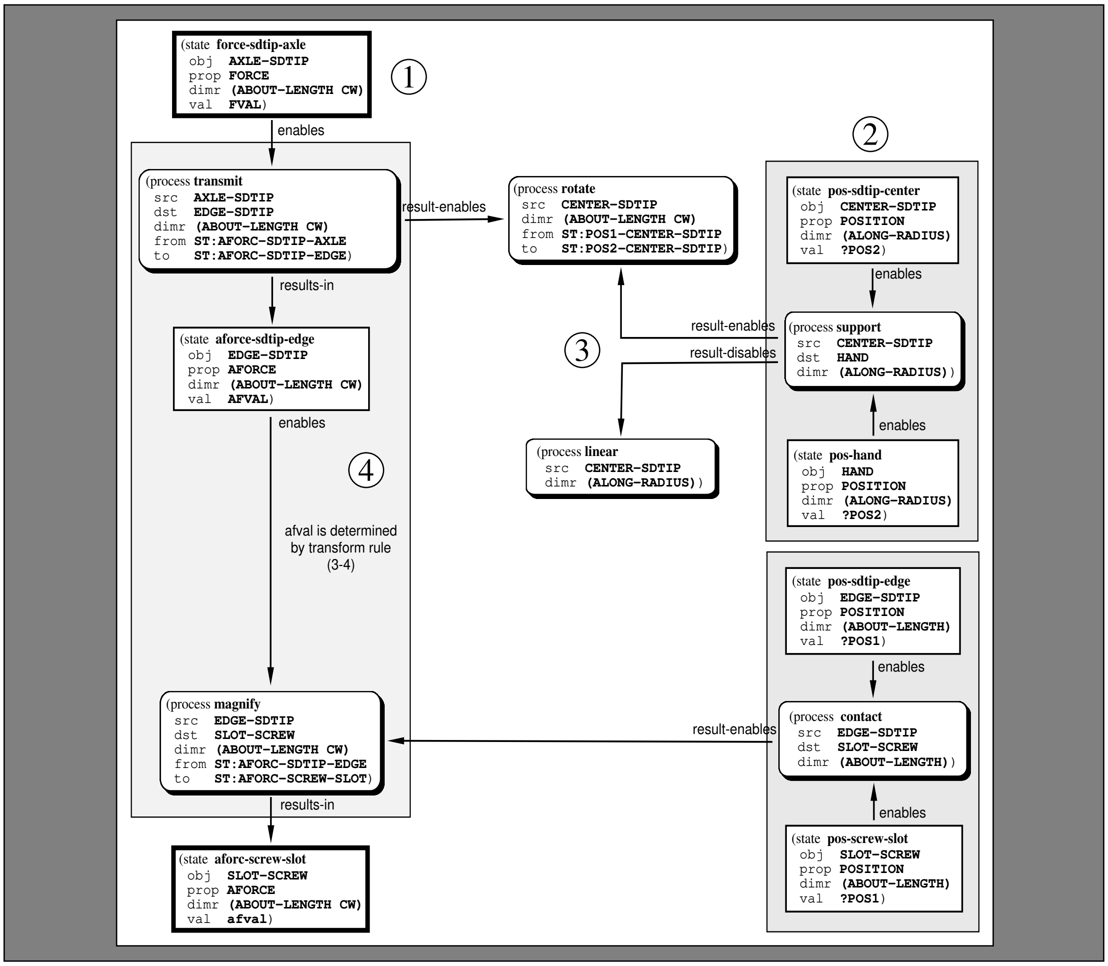


Figure 3.33 Frame representation for the screwdriver tip dynamics when the applied force is torsional.

The BPP sequences for the screwdriver components in these figures are dependent on the component statics and on the applied forces on the components, so they represent general patterns which can be used to describe the role of these components in other devices similarly configured. The shaded sequences on the left hand side of these figures represent component functions in FONM, while those on the right represent the supporting connectivity between components.

3.4 Low-Level Device Representations

This section presents the low-level dynamic representations of the 6 devices whose static and low-level dynamic representations were presented at the end of Chapter 2. The four device types presented: (1) simple devices, (2) simple compound devices, (3) multiple compound devices, and (3) complex devices are revisited with an emphasis on their low-level dynamics. The screwdriver, for example, is a simple device. Statically it is simple because all of its components are rigidly connected. Behaviorally it is simple because all of its components move together, so the behavior of the device is the same as the behavior the most specialized component.

Only the behavior for the intended function of a device is presented in the following sections. The value of these diagrams is in showing the complexity of mechanical device behavior while showing the similarities between devices provided by FONM behavioral primitives. The device representations will take the form of the first example in this section. An idealized illustration of a possible device geometry, based on FONM primitive objects, will be shown, along with the static representation diagram for the primitives. These will be followed by the behavioral sequence diagrams for each component of the idealized device pictured. Four caveats should be noted before presenting the devices:

Components of a device's static representation will only be presented when they directly affect the component's behavior. As such visual cues such as orientation and placement, which comprise a device's static representation, will be noticeably missing in these representations. In this discussion it is assumed that the FONM static representations resolve these issues. For example, the linkage-blade combinations in the fingernail clipper show a force being transmitted from along-depth of the linkage-object to along-length of the blade-object. This is made possible by the static structure, placement, which orients the axes of device components. In this device, the placement of the linkage-object and blade-object specifies that the depth dimension of the former is equivalent to the length dimension of the latter.

Second, there will also be an incomplete correspondence between the connection type illustrated in device SDDs and the associated RESTRAIN process in the behavioral sequence. This is for space considerations only. The dimension specified in the behavioral sequence will identify the most obvious dimension of the associated RESTRAIN process.

Third, some devices have specialized regions which are not depicted in the behavioral sequence. The most simplified version of the device is presented for space considerations, but alternatives will be specifically mentioned

Finally, the values associated with device behavioral states (position, aforce, iforce, and size) are left as variables throughout. Where appropriate, FONM behavioral rules which are used to specify values are noted, but otherwise value is assumed to be part of an actual simulation.

3.4.1 Simple Devices

The low-level dynamic representations of two simple devices: (1) a bottle opener, and (2) a screwdriver, are presented in this section. Each device is comprised of components which are rigidly connected, or regions of a single component. The low-level dynamic representations for three additional devices: (3) a carving knife, (4) a shovel, and (5) a spoon are presented in Appendix A.1.

Bottle Opener

A bottle opener is designed to open cans using one end and to open bottle caps at the other. Both ends make use of mechanical advantage, but one end is pointed so the magnified force is used to puncture. An idealized bottle opener is represented as two lever-objects and a blade-object, as shown in Fig. 3.34 (redrawn from Figs. 2.58, 2.59) below.

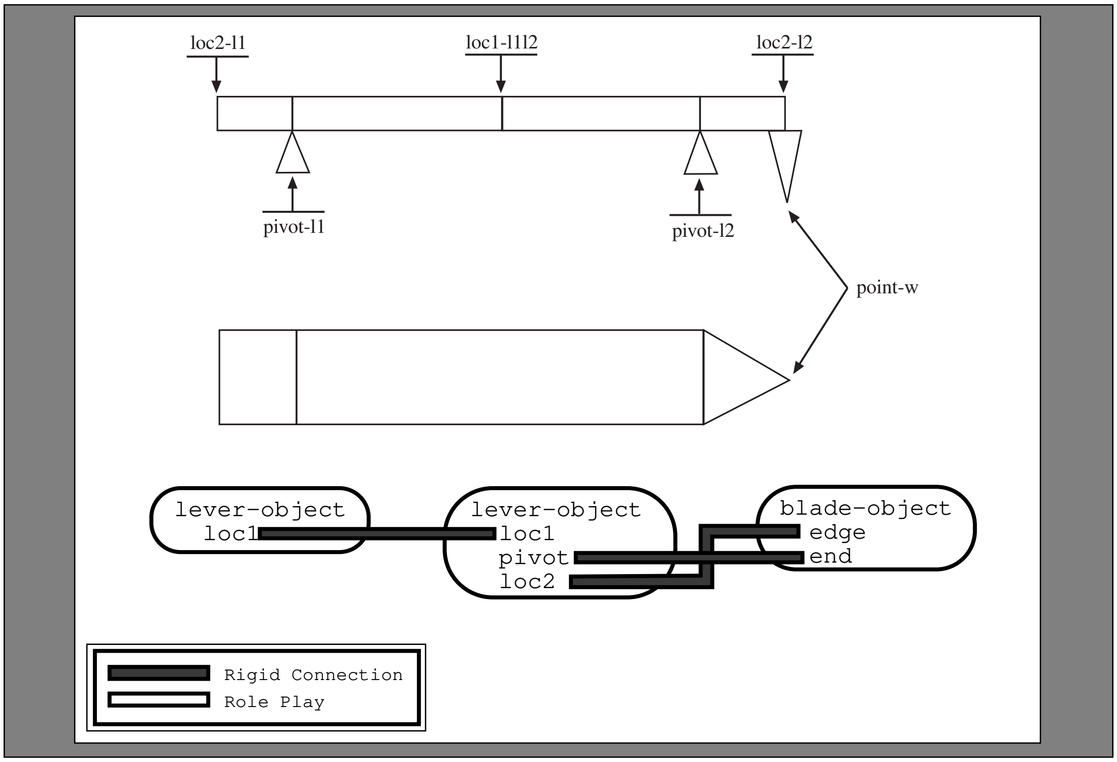


Figure 3.34 Bottle opener object primitives and static device diagram

The low-level dynamic representation for the bottle opener depicted in Fig. 3.34 is shown fleshed out in Figs. 3.35 - 3.37, one diagram for each component's behavior in the can-opening function. Fig. 3.35 illustrates the behavior of the handle (i.e., the blunt end in can opening). The effect of holding the can opener is to apply a force to the end of the device. Since the handle is represented with a lever-object that degenerates to a linkage-object, the force which initiates the behavior is shown at (1). The function of the handle is simply to transmit the lateral force to the other end (`LOC1-LEVER2`, at 2), which is enabled by the rigid connection between the two components (3). The transmission of force and the connection both enable linear motion of `LEVER2` (4). The representation for `LEVER2` behavior is illustrated in Fig. 3.36. `LEVER2` transmits force to the can lip and to the upper lid. In so doing mechanical advantage is generated in the blade end of the can opener. Since the representation makes the blade-object an explicit component, the magnified force is transmitted to the end of the blade-object. The behavior is enabled by the applied force at `PIVOT-LEVER2`. In FONM, since no internal effects are represented, any applied force which is effected at one location is transmitted to other locations. The applied force which is transmitted from the handle to `LOC1-LEVER2` is thus transmitted directly to `PIVOT-LEVER2` (1). The pivot contact between `PIVOT-LEVER2` and the substrate region2 (the can lip) en-

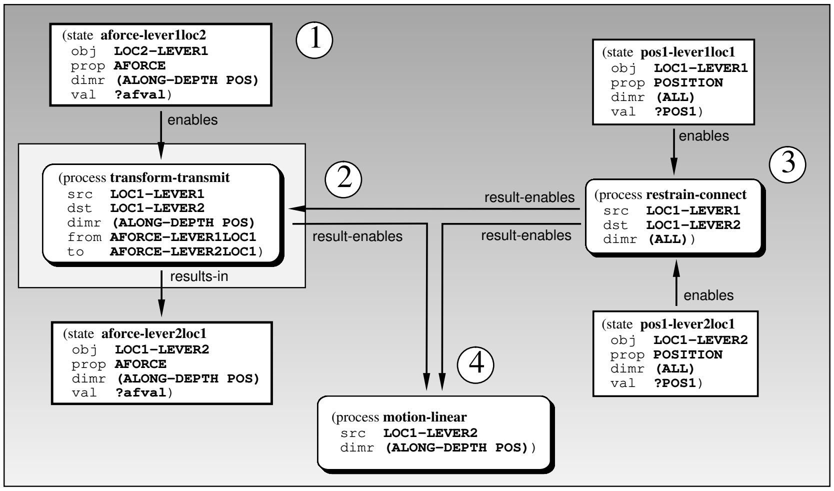


Figure 3.35 Behavioral sequence for bottle opener handle in the can-opening function.

ables the transmission of force to the can lip. It also enables rotation since the contact point

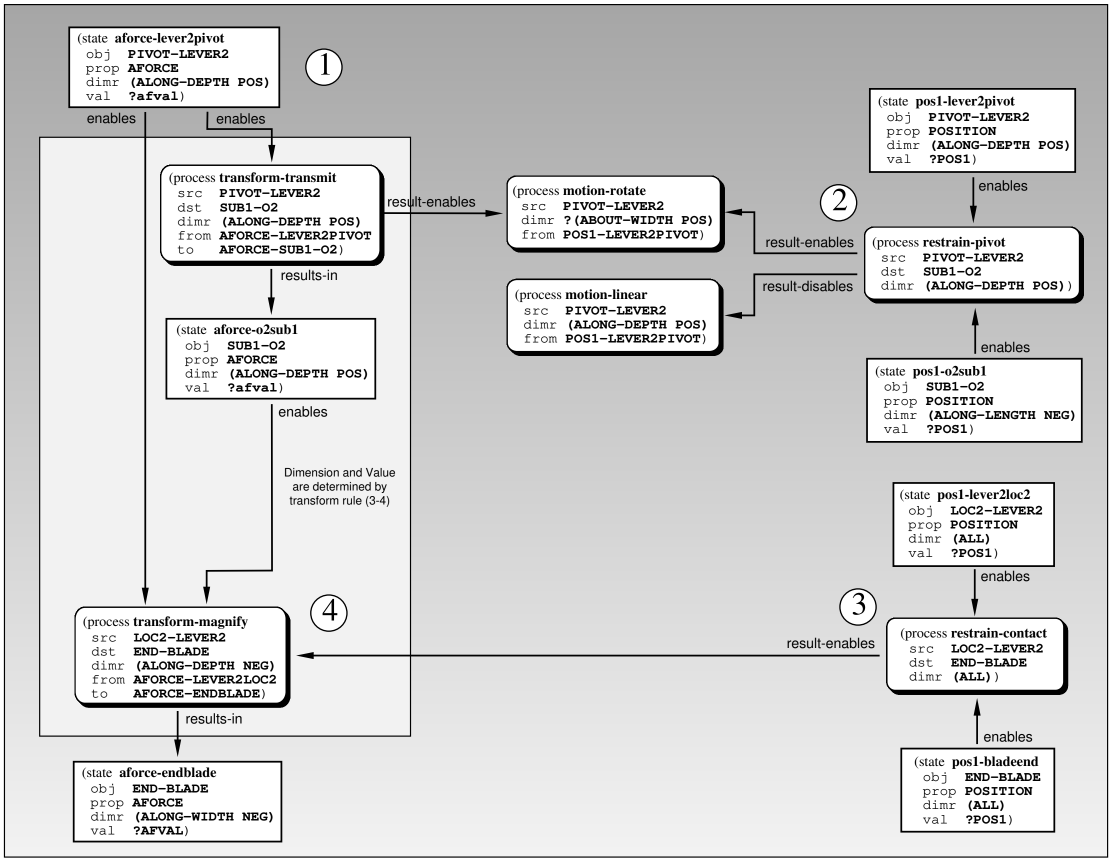


Figure 3.36 Behavioral sequence for bottle opener lever in the can-opening function.

is below the substrate region (2). The connection between LOC2−LEVER2 and the end of the blade-object (END−BLADE, at 3) enables the magnification of force to the blade according to the rules of mechanical advantage (3-4 in particular). The behavior for the blade is shown in Fig. 3.37, starting with the applied force at the end of the blade-object (1).

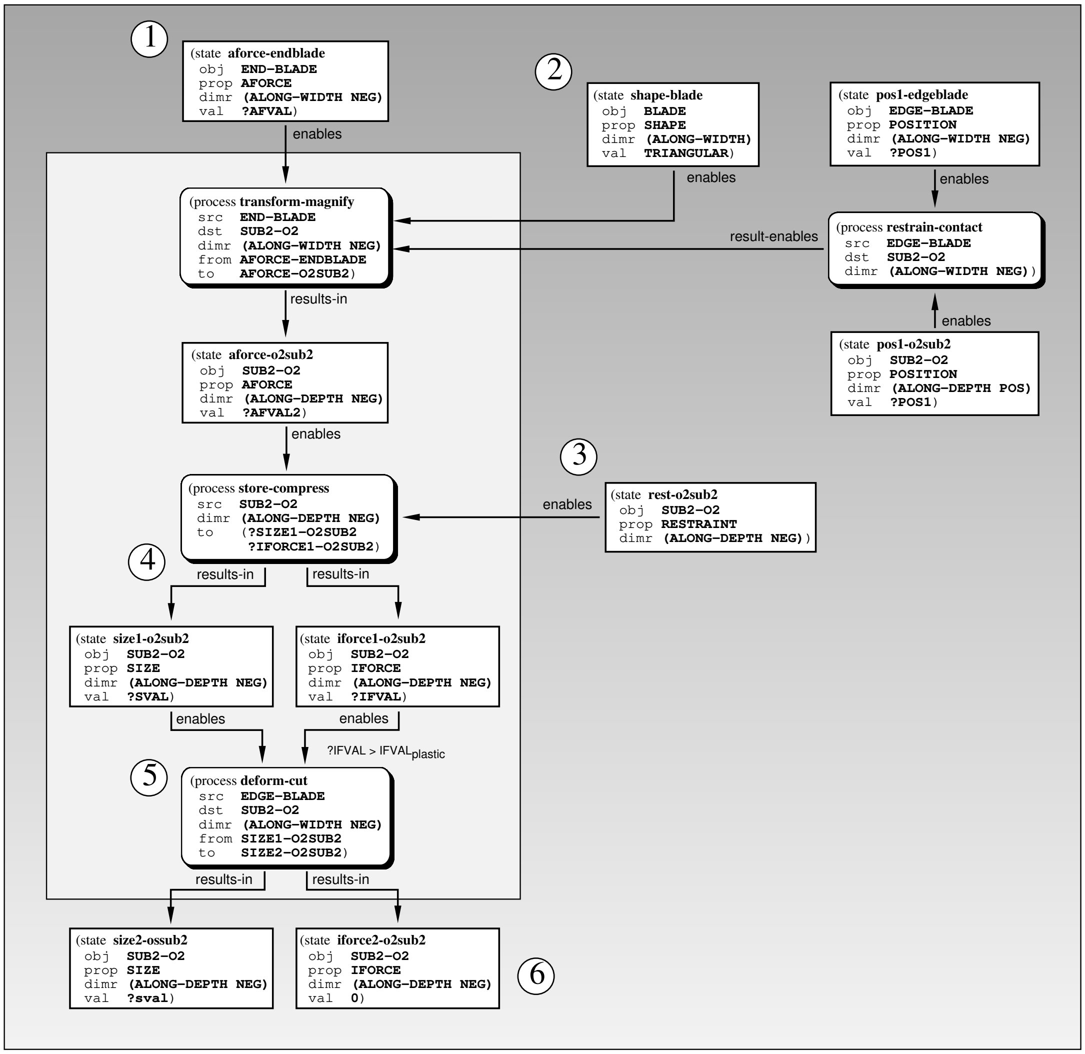


Figure 3.37 Behavioral sequence for bottle opener blade in the can-opening function.

Linear magnification is effected in the blade-object as a result of its triangular shape (2), its contact with the can lid, and the applied force at its end. It is assumed in this figure that the can is restrained along its length (3), although the representation of such would be a part of a pragmatic description of use. The combined result is a compression of the lid (`SUB2-O2` at 4). Assuming that the force exceeds the plastic limit of the can lid ($\mathtt{IFVAL}_{\mathtt{plastic}}$), the lid is cut (5), which changes its size and releases the stored energy (6).

Screwdriver

The idealized screwdriver and its low-level dynamic representations are shown in Figs. 3.30 - 3.33. The idealize screwdriver and its static device diagram are depicted in Fig. 3.38

for continuity and will not be discussed further in this section.

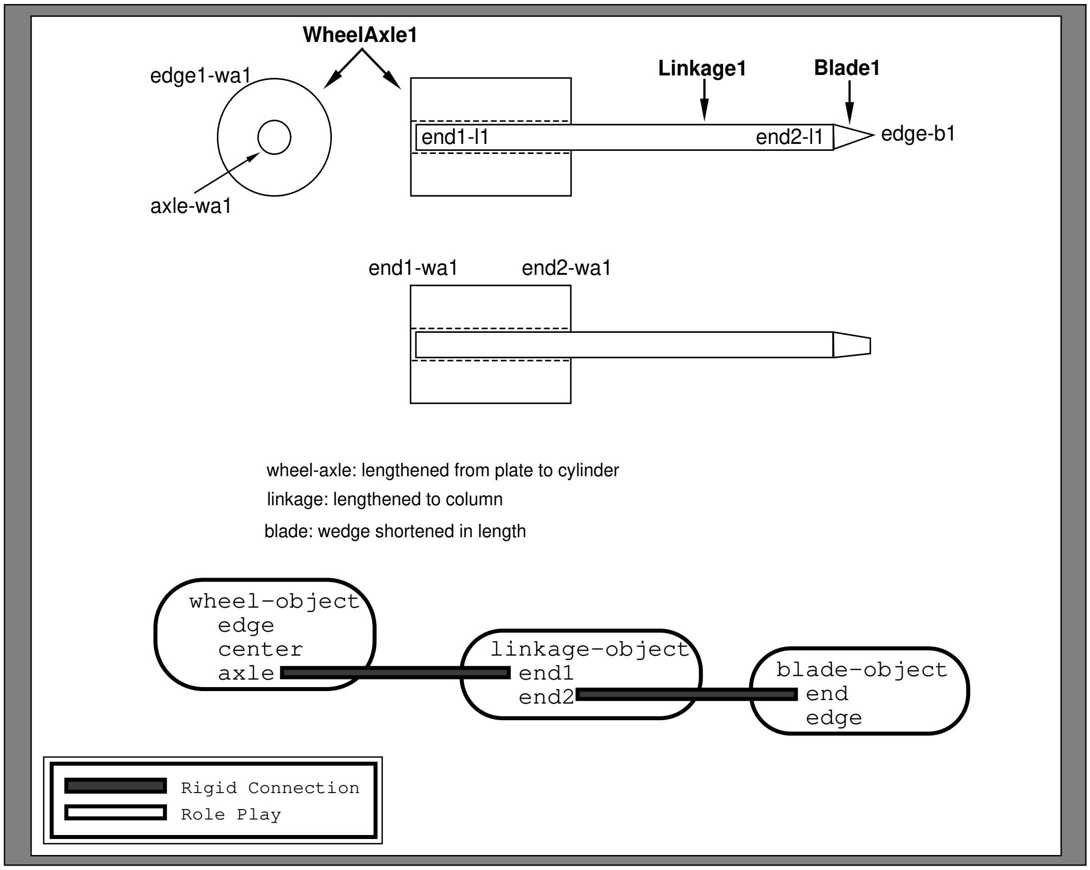


Figure 3.38 Screwdriver object primitives and static device diagram.

3.4.2 Simple Compound Devices

Devices in which relative motion between components is possible are *compound* devices. Compound device dynamics are representationally significant because components can behave independently. The behavioral representations of simple compound devices show striking similarities, because the types of relative motions for simple devices is limited. In this section, two devices: (1) a nutcracker, and (2) a clothes pin are presented.

Nutcracker

The intended function of a nutcracker is to restrain the motion of and crack the shell of a nut. The idealized nutcracker and its static device diagram are illustrated in Fig. 3.39.

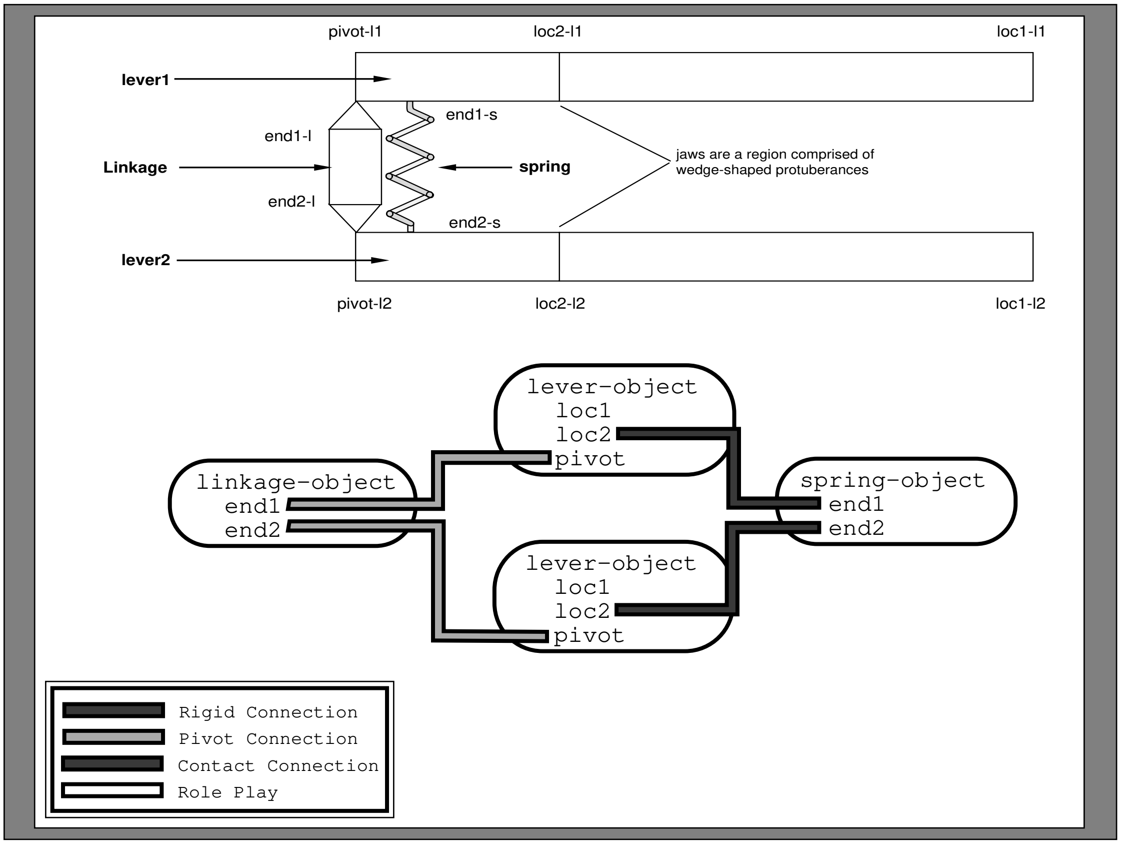


Figure 3.39 Nutcracker object primitives and static device diagram.

The behavioral sequences for the idealized nutcracker components in the cracking function are illustrated in Figs. 3.39 - 3.43. The device behavior is enabled by pressing the handles together. The applied forces are transmitted from the handles to the pivot location (at 1 in

Figs. 3.39, 3.41), as mediated by the connection labeled (2). The right side of the diagram

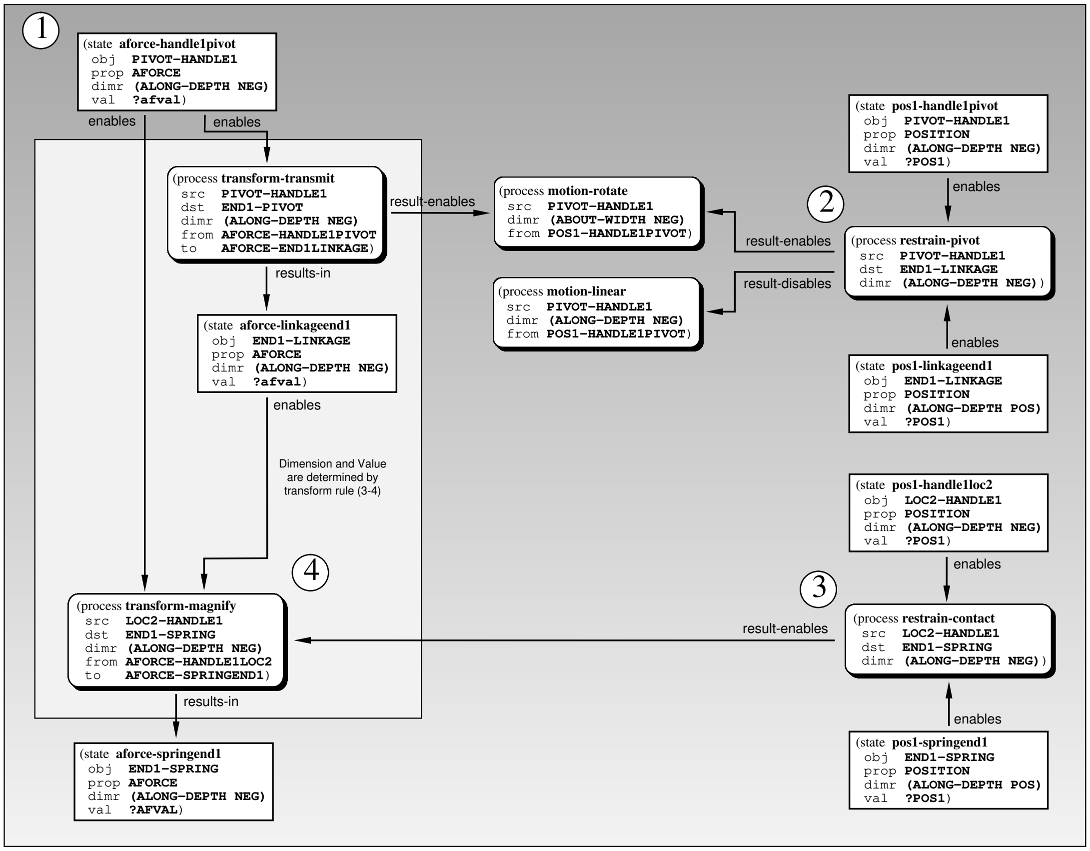


Figure 3.40 Behavioral sequence for upper nutcracker handle in the cracking function.

is similar to `LEVER2` of the bottle opener, since two contacts are needed to produce mechanical advantage. The difference is in the location of `LOC2-LEVER` in the bottle opener and `LOC2-HANDLE1` and `LOC2-HANDLE2` on the nutcracker, which result in the type1 and type2 leverage, respectively. The resolution of forces in the linkage-object are shown

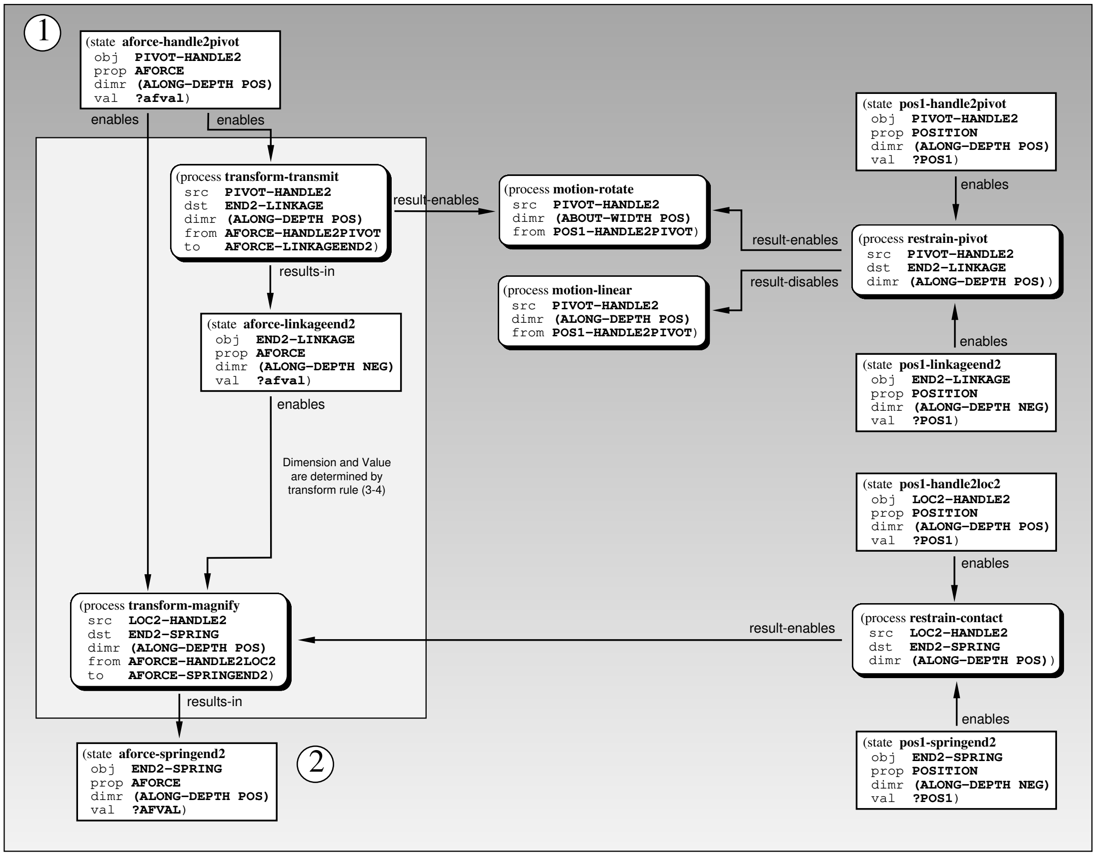


Figure 3.41 Behavioral sequence for lower nutcracker handle in the cracking function.

in Fig. 3.42. The two forces transmitted from `HANDLE1` and `HANDLE2` to the linkage-object (i.e., `LINKAGE`) are shown at (1). The force is transmitted through the linkage-object to the other lever-object (`END1-LINKAGE` → `PIVOT-HANDLE2`, and `END2-LINKAGE` → `PIVOT-HANDLE1`, at 2). The result is forces of equal value and opposite direction, which cancel (at 4), and which collectively disable motion of the linkage-object (3). The

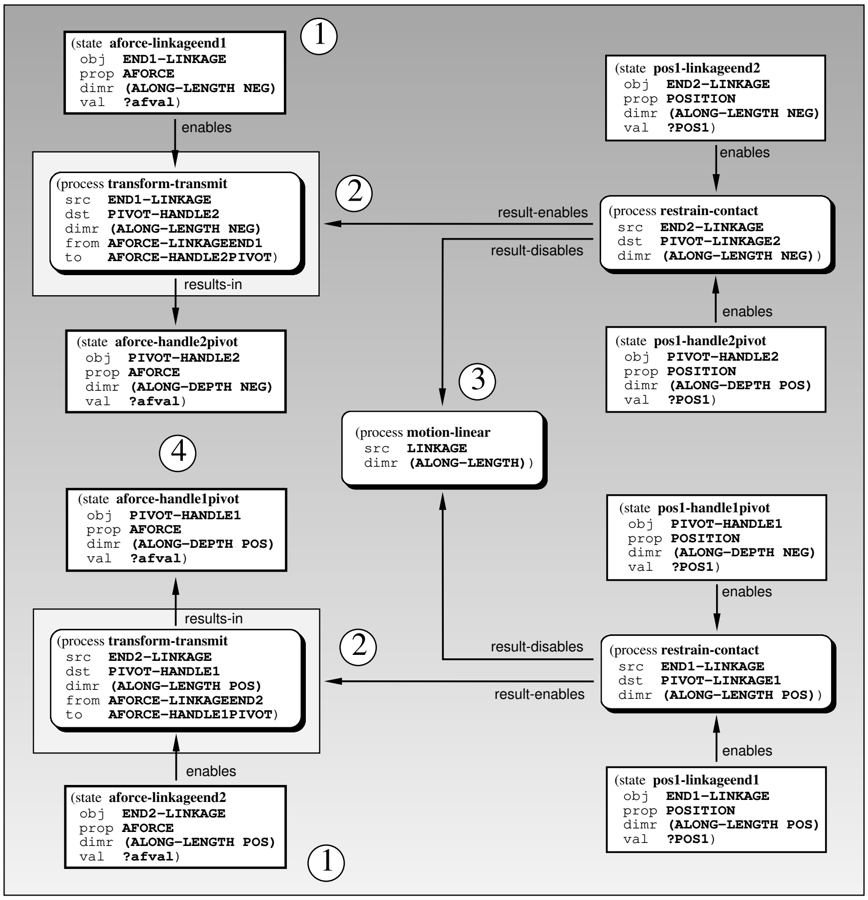


Figure 3.42 Behavioral sequence for nutcracker pivot in the cracking function.

final figure, for the spring-object, is shown in Fig. 3.43. The force which is transmitted from the handles in Figs. 3.39, 3.41 is shown at (1). For the spring-object to compress, its motion must be restrained. The contact between the *other* spring-object location and the *other* lever-object (`END2–SPRING` with `LOC2–HANDLE2`, for `END1–SPRING`, and

`END1-SPRING` with `LOC2-HANDLE1`, for `END2-SPRING`, at 2) achieves this effect (3).

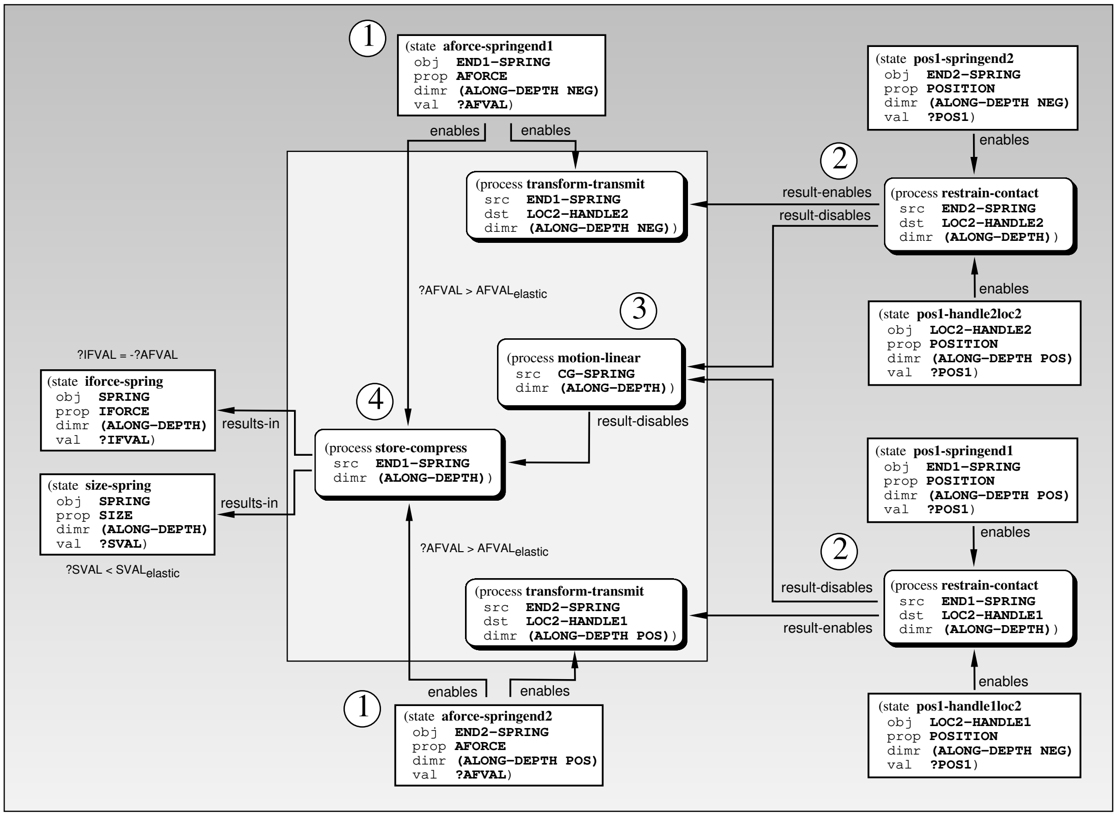


Figure 3.43 Behavioral sequence for nutcracker spring in the cracking function.

The combination of applied force which exceeds the elastic threshold (4), and motion restraint, enables spring-object compression (4). To add the effect of containment in the nutcracker to these representations, two container-objects (or container-object combinations, as presented in Chapter 2, Page 88) could be added at `LOC2-HANDLE1` and `LOC2-HANDLE2`. Two more behavioral diagrams would then be required.

Clothes Pin

A clothes pin holds objects together without manual control. The device is comprised of three components: two lever-objects and a spring-object. The spring-object is employed to keep the lever-objects together, as well as to keep the jaws closed, as shown in the idealized clothes pin and static device diagram depicted in Fig. 3.44.

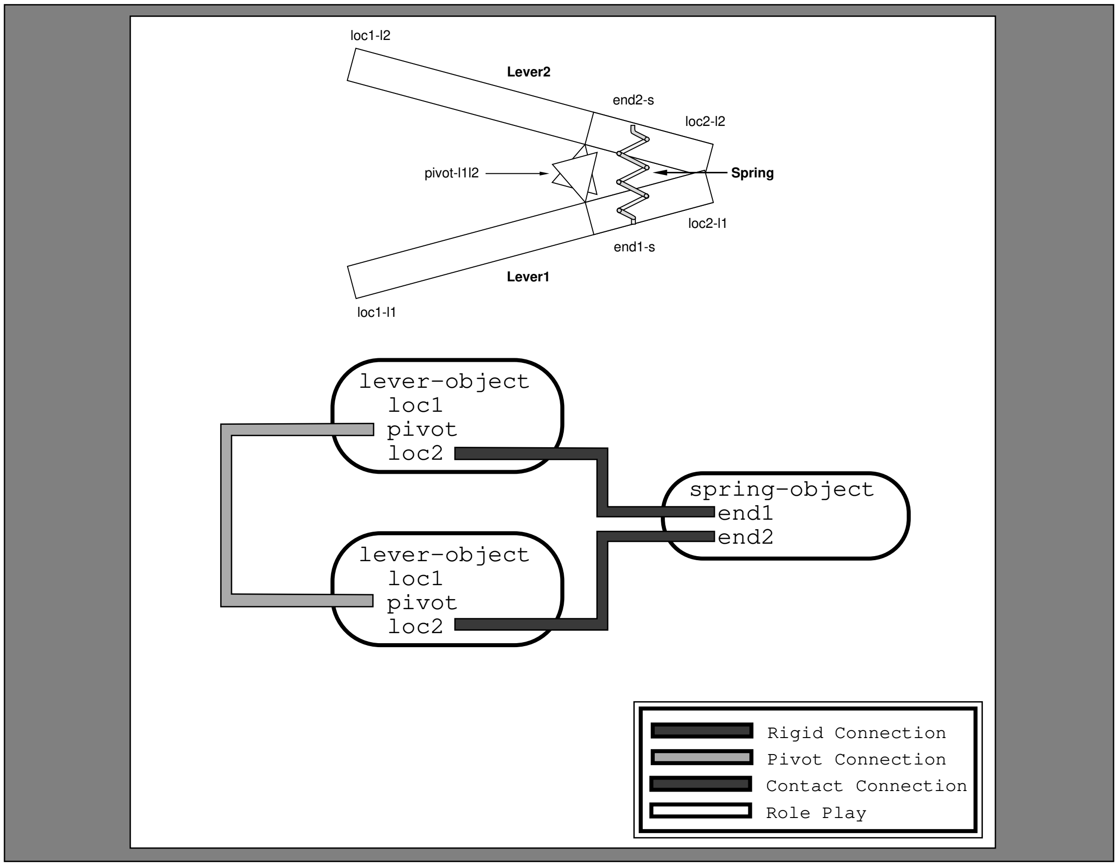


Figure 3.44 Clothes pin object primitives and static device diagram.

The static representation diagram for the clothes pin is almost identical to that of the nutcracker in Fig. 3.39, but they behave differently under the same applied force because the lever-object pivot region is located away from the handle ends (it was at the handle ends in the nutcracker). The behavioral sequences for the idealized clothes pin components are shown for the clasping function in Figs. 3.44 - 3.47. The location of the clothes pin spring-object and the orientation of its contact (i.e., above) the handles results in the clothes pin being closed at rest, whereas the nutcracker is open at rest. In other respects the behavioral representations for the devices are identical. Figs. 3.44, 3.46 show the magnification of force to the spring-object. In each figure, the applied force is shown at the lever-object pivot

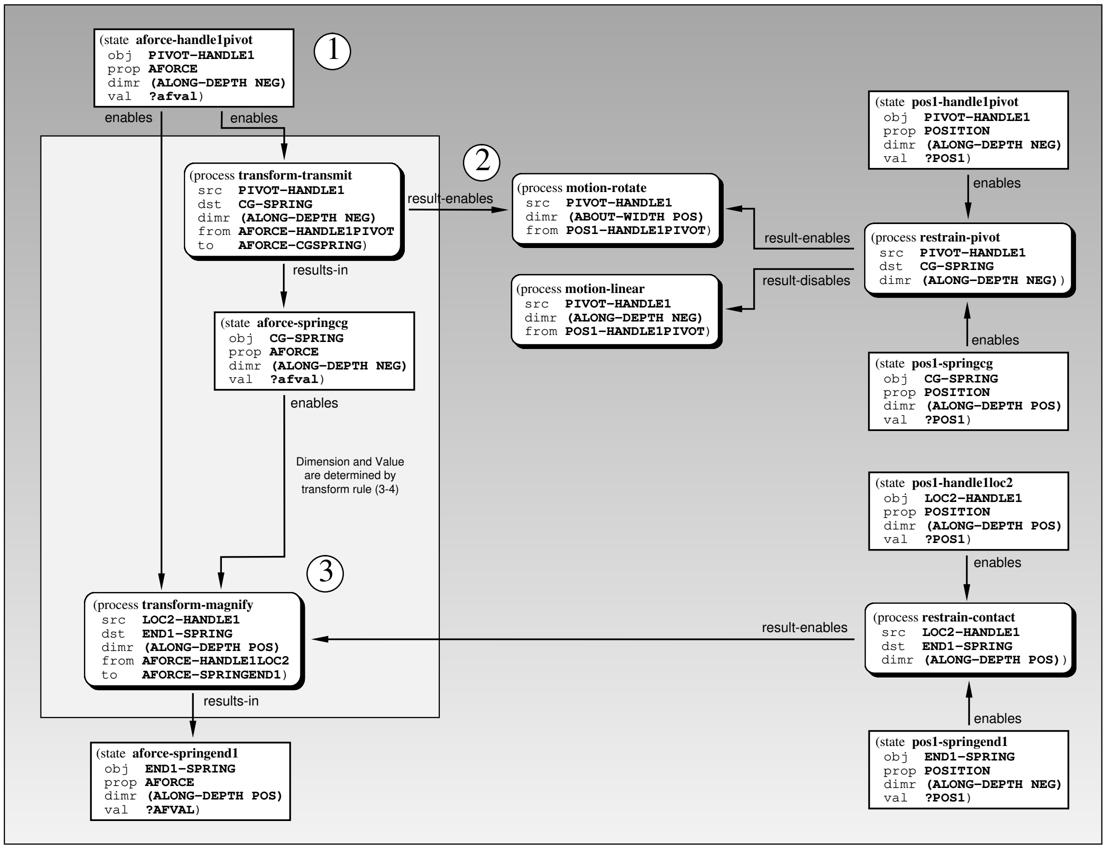


Figure 3.45 Behavioral sequence for upper clothes pin handle in the clasping function.

region (1). The transmission of force at `PIVOT−HANDLE1`, and `PIVOT−HANDLE2`, to `CG−SPRING`, is interesting, since the spring-object is effectively the fulcrum about which the lever-objects rotate. The magnification of force (3), thus works on two locations on the spring-object. The spring-object behavioral sequence is illustrated in Fig. 3.47. The spring

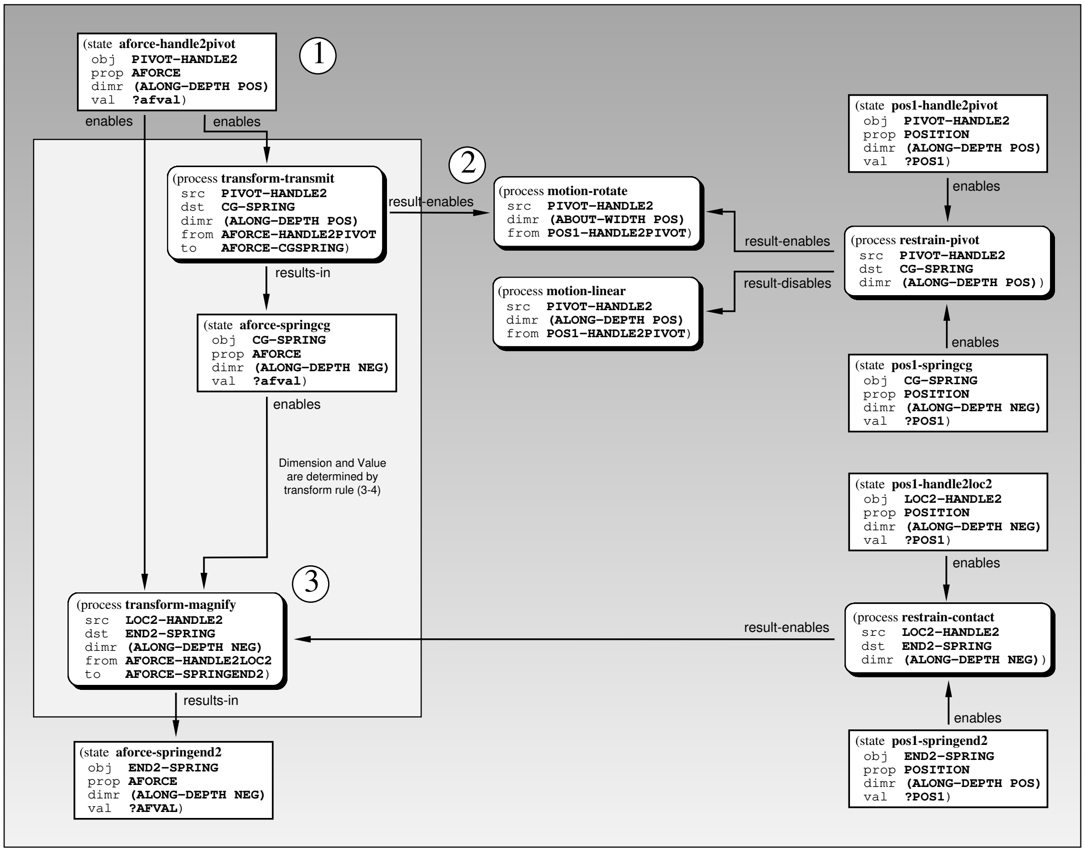


Figure 3.46 Behavioral sequence for lower clothes pin handle in the clasping function.

in the clothes pin is similar to that in the nutcracker inasmuch as both springs are connected to two handles which serve as levers. The detail shown in Fig. 3.43 for the forces propagating from the lever-objects to the spring-object has been omitted in Fig. 3.47. Only one half has been shown, from the upper handle to the spring CG.

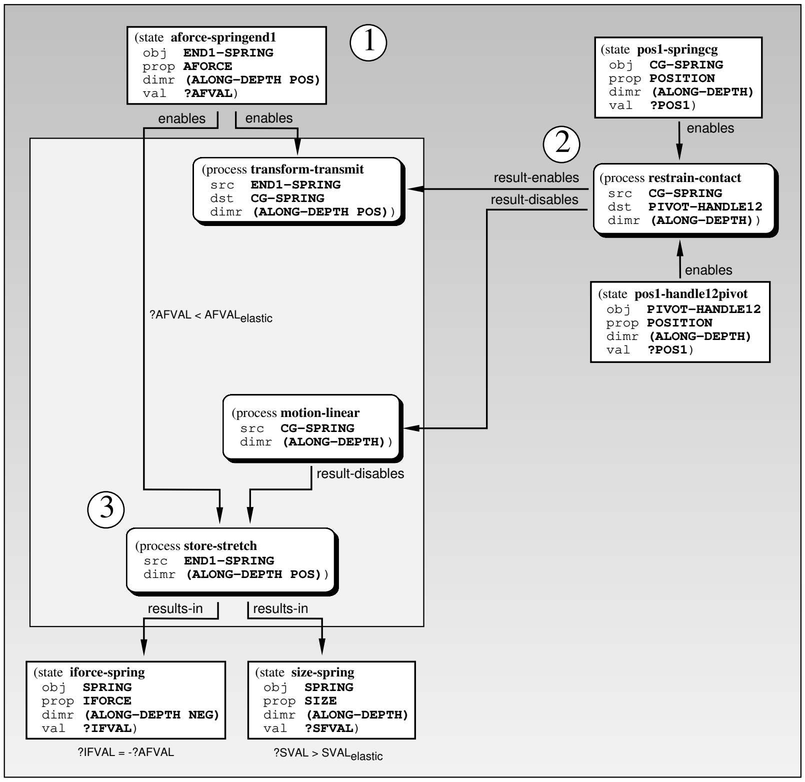


Figure 3.47 Behavioral sequence for clothes pin spring in the clasping function.

The other half is identical in form (see Fig. 3.43) to the nutcracker spring-object.

3.4.3 Multiple Compound Devices

Where a clothes pin has two components which can move independently, a multiple-compound device has more than one independently moving component. Although the statics and intra-object dynamics are fairly straightforward, the inter-object dynamics begins to become complex when fully elaborated. One such device is the fingernail clipper, whose behavioral representation is presented in this section.

Fingernail Clipper

A fingernail clipper clips nails by forcing two blades together. The force necessary to clip is achieved with a lever-object through magnification. The magnification results from a pivot connection between the lever-object and a pin (a linkage-object) which is restrained from motion by the lower linkage/blade. The lever-object has a fulcrum on it which presses against the upper linkage/blade. The magnified lateral force on the upper linkage/blade deflects it like a spring, forcing the two blade-objects together. An idealized fingernail clipper and its associated static device diagram are shown in Fig. 3.48. The linkage/blade objects are represented as separate objects rigidly connected. The associated linkage-objects are represented as being pivoted, and their function as springs is represented with a spring-object. The handle (lever-object) can move independently of the linkage/blade-objects, and the blade-objects can move with respect to each other. The handle is pivoted to the pin (at

1 in Fig. 3.48), so it can rotate about the pin in two radial dimensions.

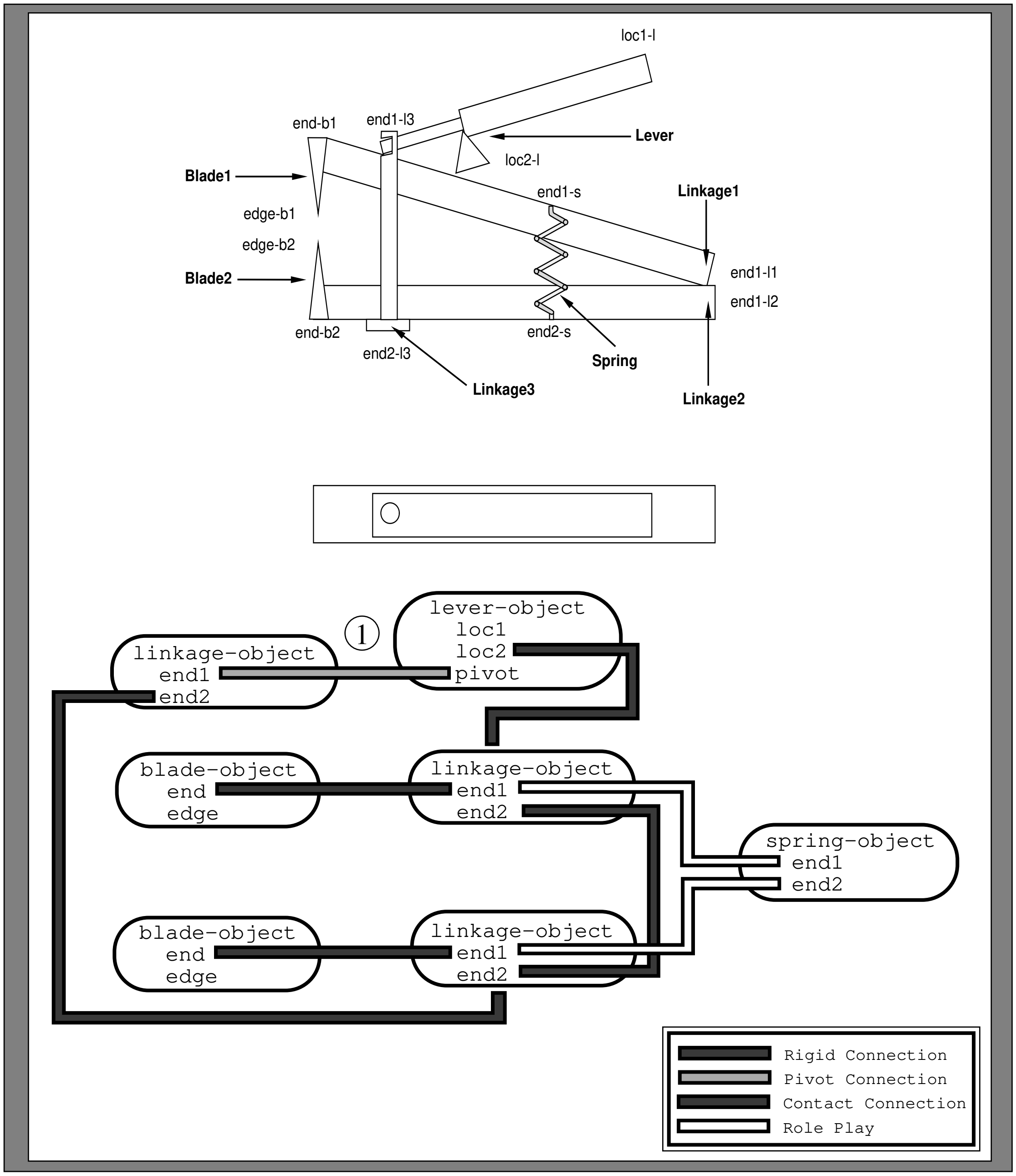


Figure 3.48 Fingernail clipper object primitives and static device diagram.

The behavioral sequences of the fingernail clipper components as they are used in the clipping function are shown in Figs. 3.49 - 3.55.

Fig. 3.49 shows the behavioral sequence associated with the fingernail clipper handle. The

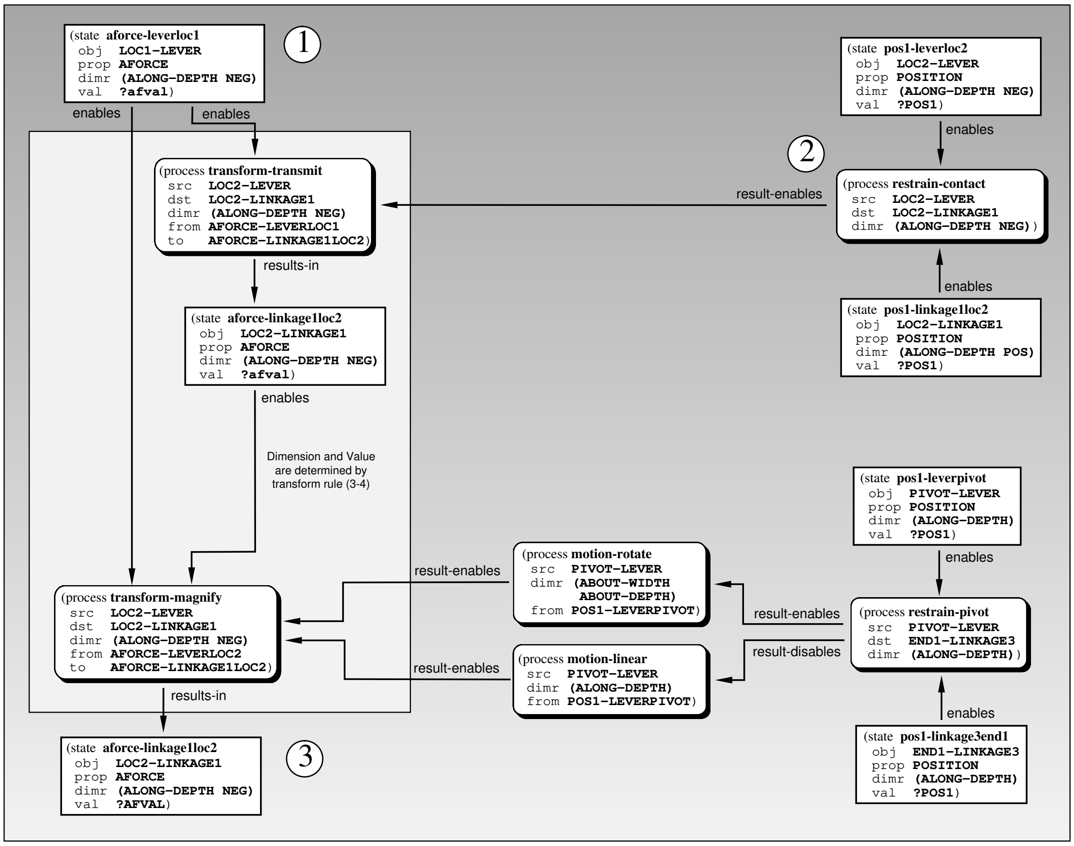


Figure 3.49 Behavioral sequence for fingernail clipper handle in the clipping function.

force resulting from pressing the handle (1) is transmitted to the upper linkage (LOC2-LINKAGE1) as a result of the contact shown at (2). The pivot relationship with the pin (END1-LINKAGE3) enables the force magnification (3).

The upper linkage-object (LINKAGE1) and blade-object (BLADE1) behaviors are shown in Fig. 3.50 and Fig. 3.51. The upper linkage is connected to three components: `SPRING` (1), `LINKAGE2` (2), and `BLADE1` (3), as noted by the labels in Fig. 3.50. The force which is transmitted from `LEVER` to `LINKAGE1` is transmitted to these components. Fig. 3.51 shows how the magnified force (at 1) is again magnified (i.e., linear magnification) due to

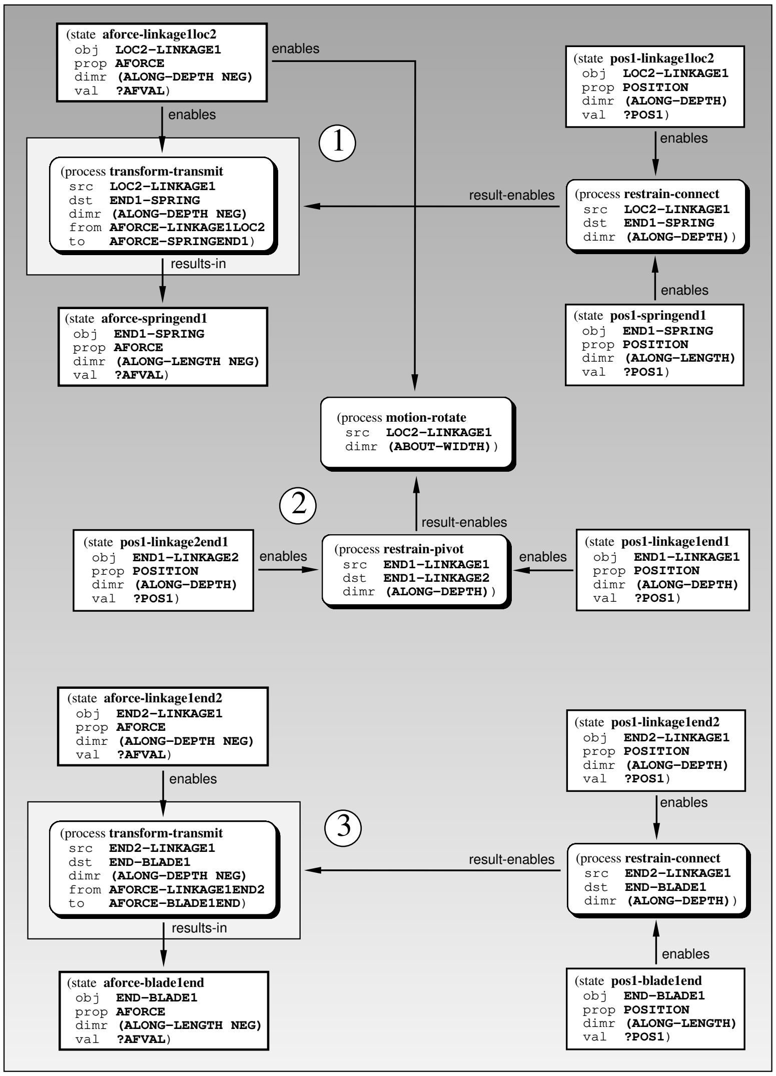


Figure 3.50 Behavioral sequence for the upper fingernail clipper linkage in the clipping function.

the shape of BLADE1 (2). The upper blade is assumed to be in contact with a fingernail

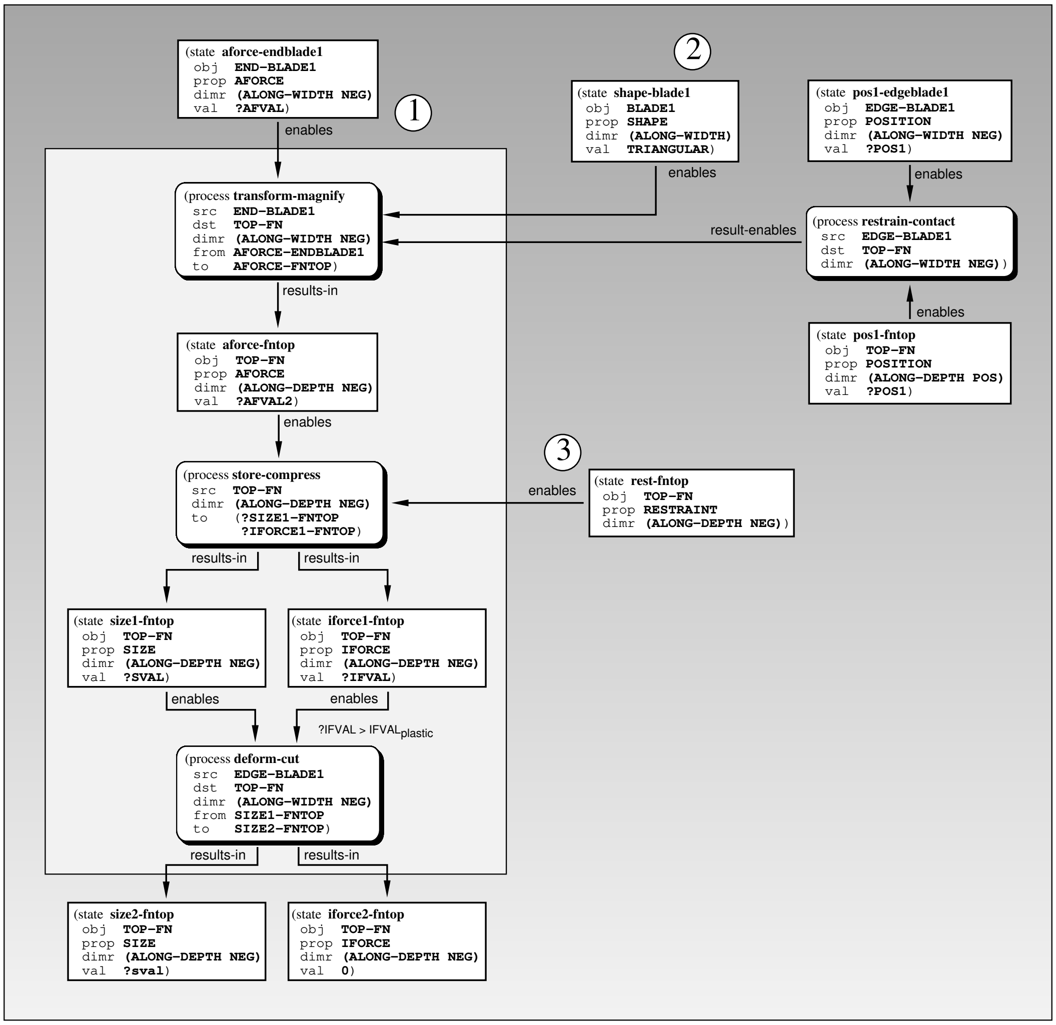


Figure 3.51 Behavioral sequence for the upper fingernail clipper blade in the clipping function.

(FN) for the example, and the fingernail is assumed to be restrained in the dimension and direction of BLADE1 movement (i.e., ALONG-DEPTH NEG, at 3). The lower linkage (LINKAGE2, Fig. 3.52) differs from LINKAGE1 inasmuch as the pin (LINKAGE3) is in contact with LINKAGE2 (at 1 in Fig. 3.55) and not with LINKAGE1, and because LEVER magnifies force to LINKAGE1, but that force is only transmitted to LINKAGE2 through LINKAGE3 (at 2 in Fig. 3.55) due to the pivot connection assumed between the upper and lower linkage-objects in the idealization. Figs. 3.53, and 3.54 are otherwise identical to

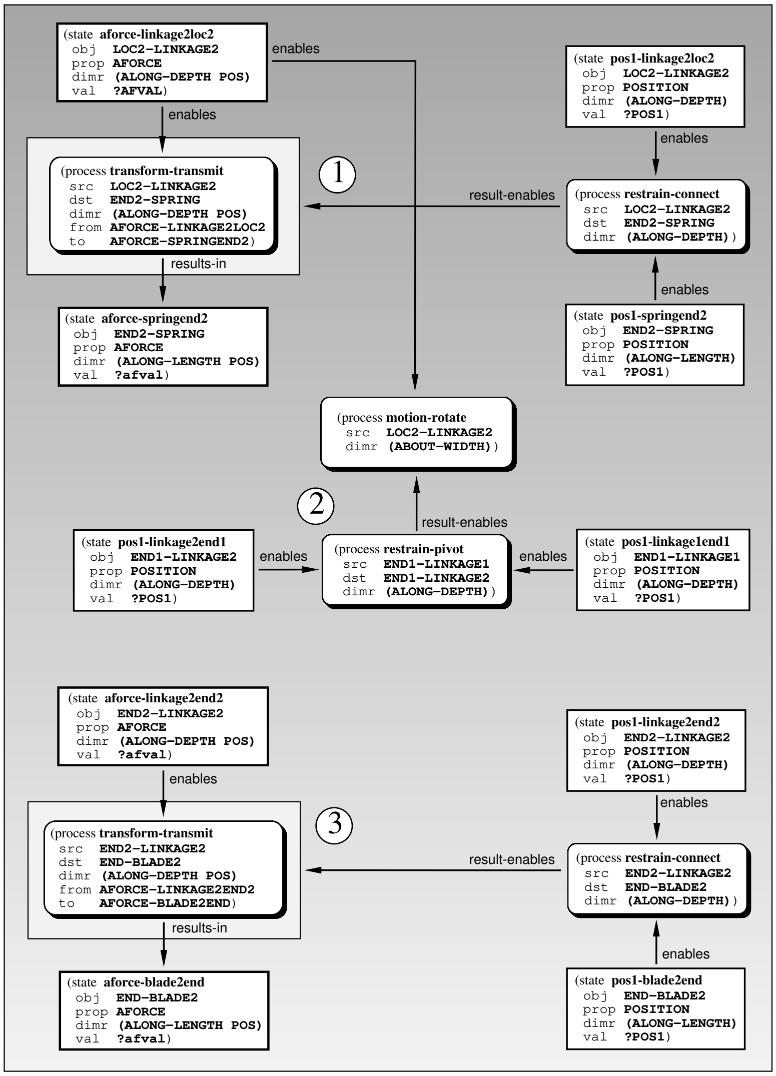


Figure 3.52 Behavioral sequence for the lower fingernail clipper linkage in the clipping function.

Figs. 3.50, and 3.52.

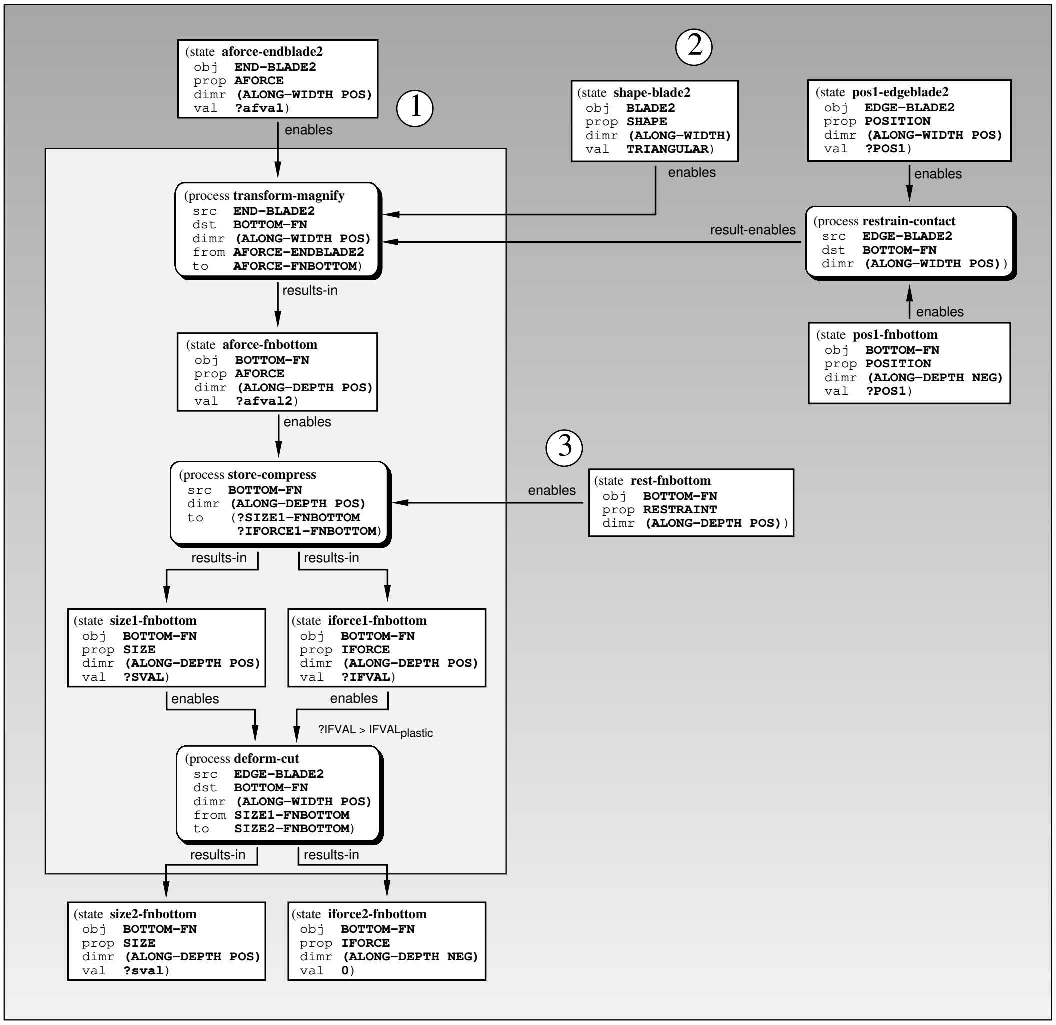


Figure 3.53 Behavioral sequence for the lower fingernail clipper blade in the clipping function.

The fingernail clipper spring-object behavior is illustrated in Fig. 3.54, and is identical in behavior to that shown for the nutcracker device in Fig. 3.43.

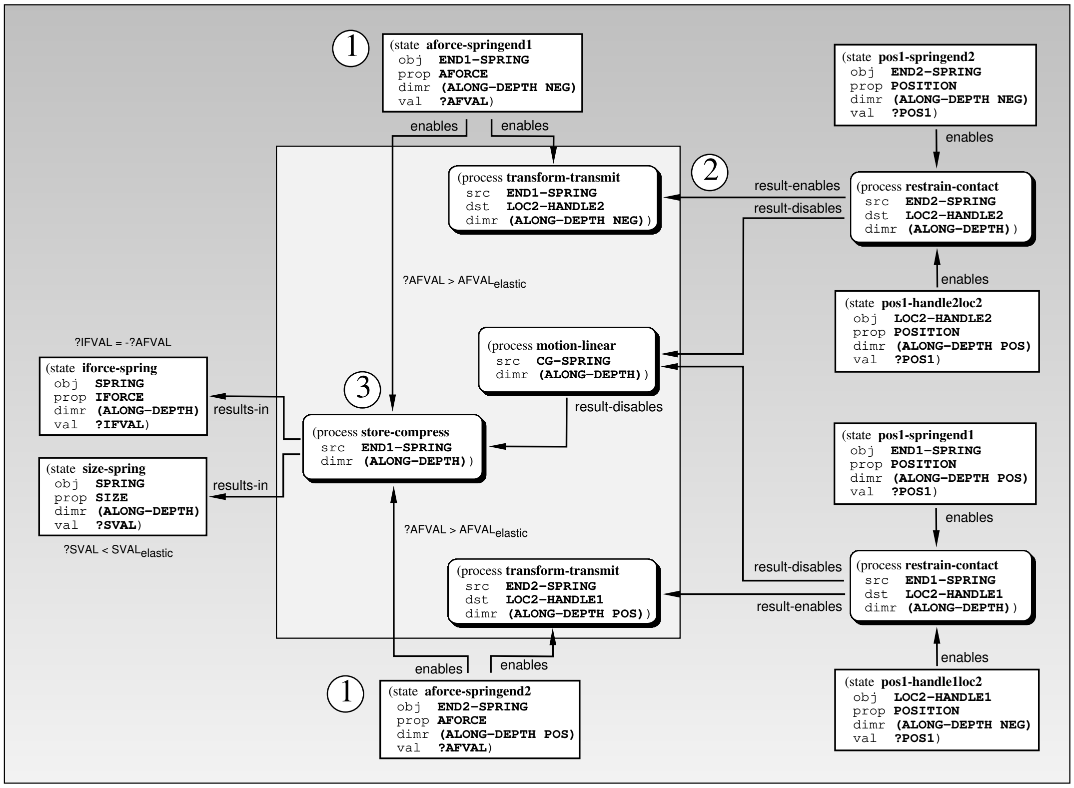


Figure 3.54 Behavioral sequence for the fingernail clipper spring in the clipping function.

The nutcracker pin is shown in Fig. 3.55.

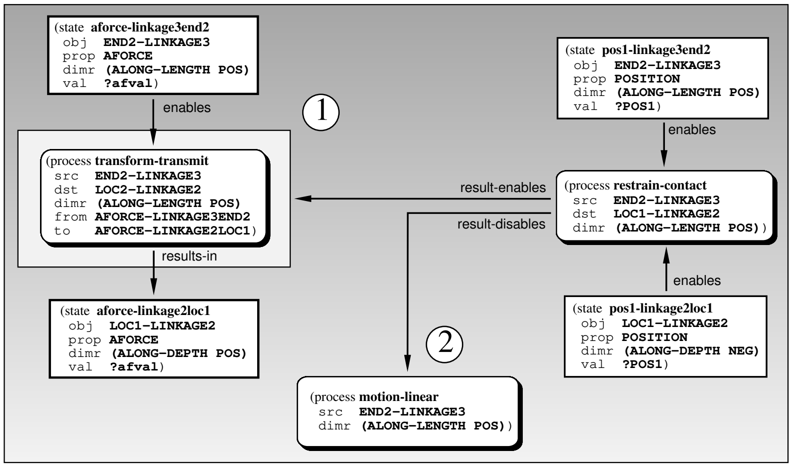


Figure 3.55 Behavioral sequence for the fingernail clipper pin in the clipping function.

3.4.4 Complex Devices

Complex devices involve objects which may or may not be permanent components of the device, have complex motions, or have large numbers of components. One such example is a toy gun, which has a projectile (dart) which is only connected to the device when being armed and preparing to fire. Another complex device is a mousetrap, since the hook and baitplate motion are both confined to the limits imposed by their connectors, but free to move in every other respect. In this section, the toy gun's behavioral representation is presented.

Toy Gun

The idealized toy gun and static device diagram are illustrated in Fig. 3.56.

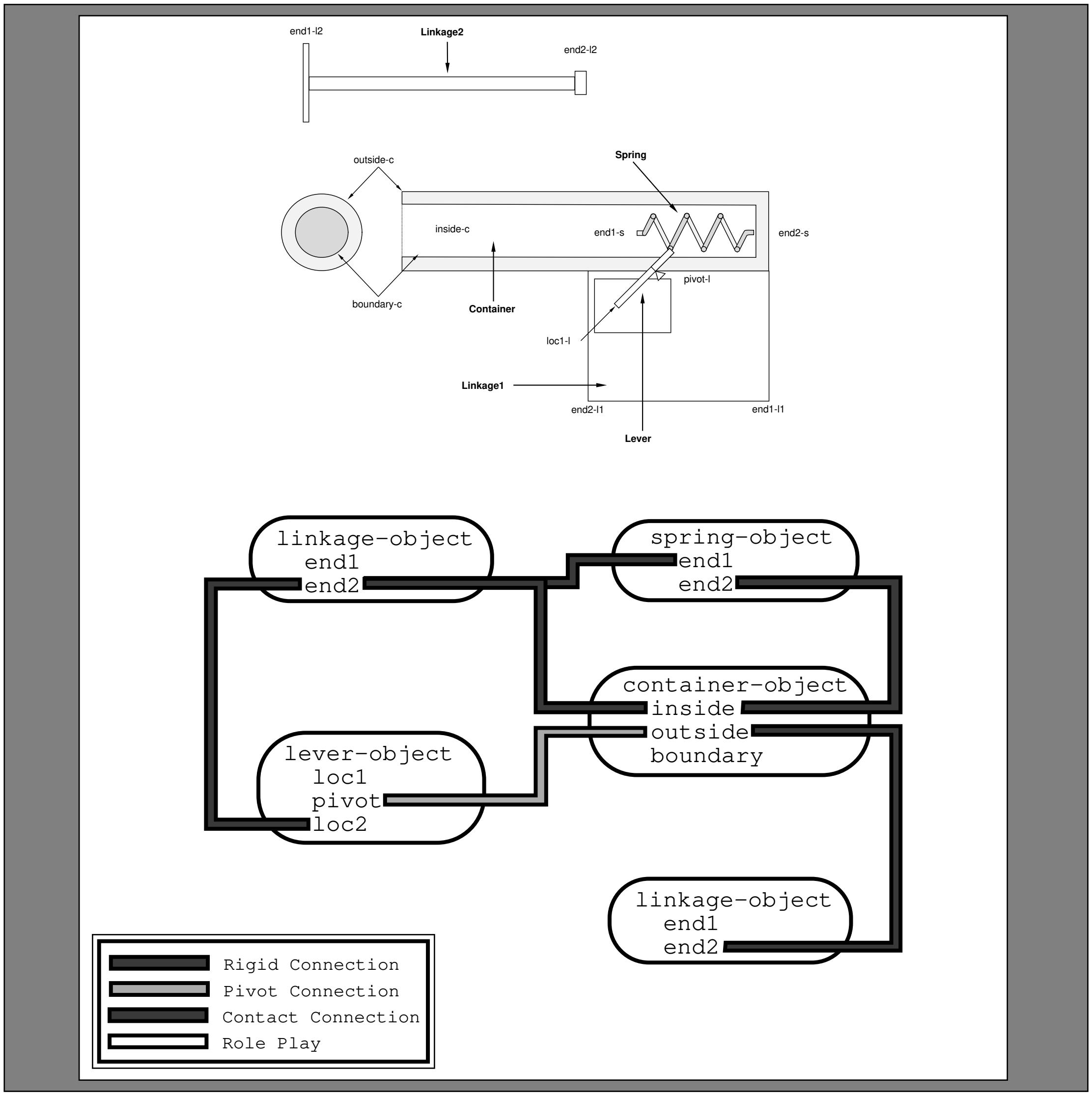


Figure 3.56 Toy gun object primitives and static device diagram.

The behavioral sequences associated with arming the toy gun are illustrated in Figs. 3.58 - 3.61. Fig. 3.58 illustrates the transmission of force from the end of the dart (`END1-LINK-`

AGE1) to the spring inside the barrel (END1-SPRING).

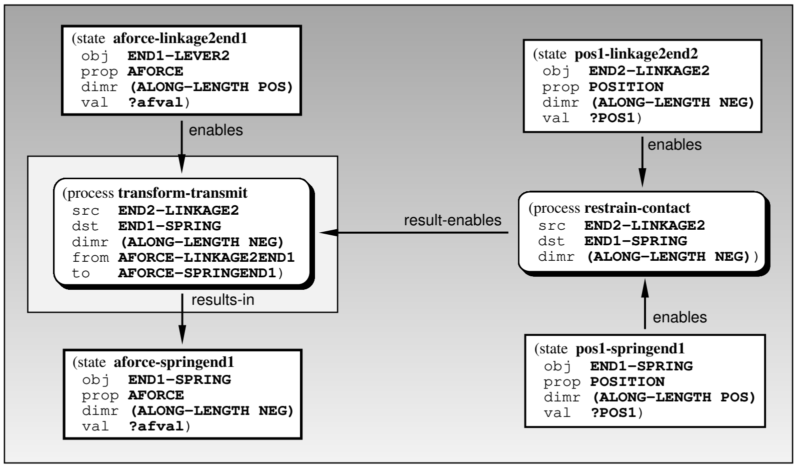


Figure 3.57 Behavioral sequence for the toy gun dart in the arming function.

Fig. 3.58 shows the handle and how force is transmitted from the handle (LINKAGE1) to the barrel (CONTAINER). An applied force at the trigger (LEVER) is assumed for the figure. The applied force is transmitted to the handle (LOC1-LINKAGE1), which is the location where the handle and barrel are connected (2). The second connection (rigid, as noted at 2), enables force transmission to the barrel (3). This connection also enables force transmission back through the barrel to the handle when the dart (LINKAGE2) is pushed against the spring-object. The barrel (`CONTAINER`) behavior is represented in Fig. 3.59. The figure is shown for the dart already contained by the barrel and represents how the container affects its ability to move. The enabling force is the applied force on the dart which resulted in its motion into the barrel. The RESTRAIN process (1), which represents the relationship between the dart and barrel radially, enables linear motion for the dart while disabling radial motion of the dart (2). The enabled linear motion results in a change in dart position which, if equivalent to `INSIDE-CONTAINER` (3), enables `RESTRAIN-CONTACT` (4) which, in turn, disables dart motion (5). The effect of `RESTRAIN-CONTAIN` is to define

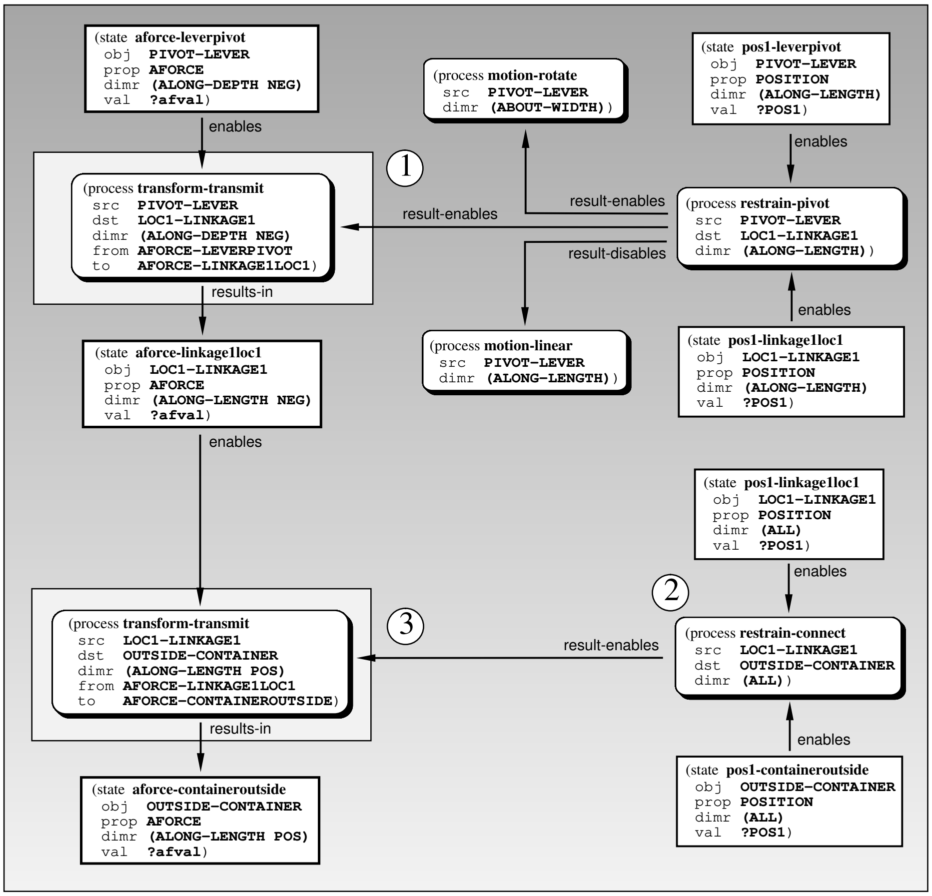


Figure 3.58 Behavioral sequence for the toy gun handle in the arming function.

the limits of motion and not to otherwise restrict, or care about, the position of the dart. The

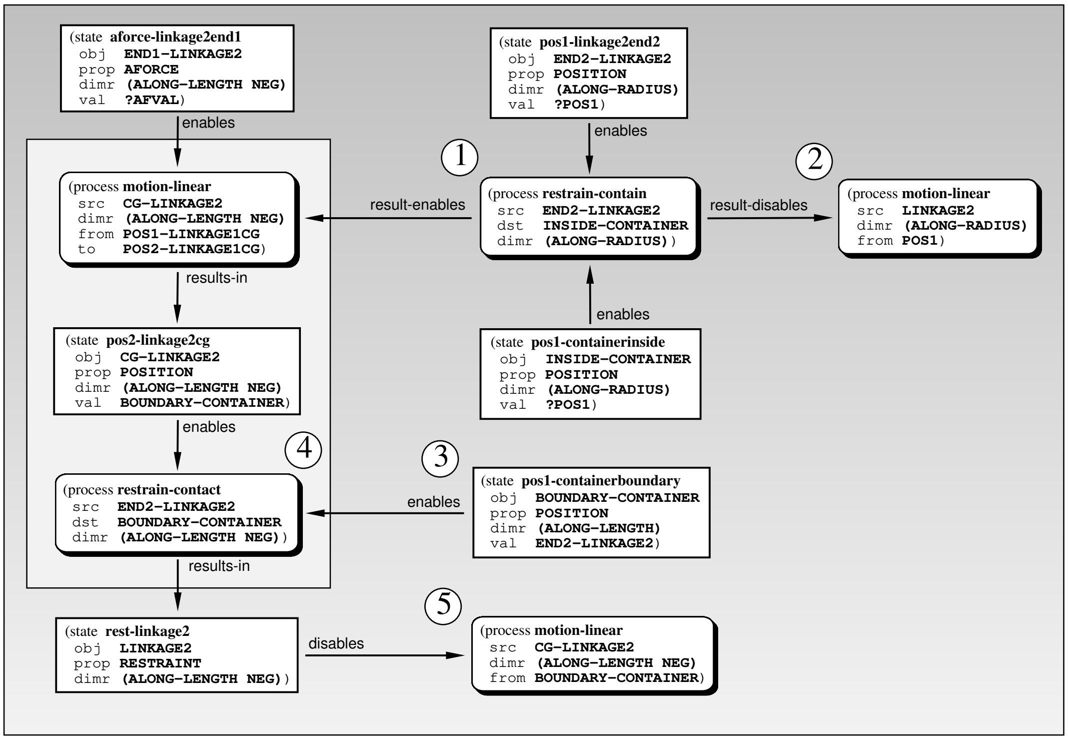


Figure 3.59 Behavioral sequence for the toy gun barrel in the arming function.

toy gun trigger mechanism is represented with a lever-object, shown in Fig. 3.60 instantiated for the dart in contact with the spring-object when the gun is armed. The behavior is enabled by pulling on the trigger, and the initial applied force in the diagram is the force at `LOC2-LEVER`. The force is transmitted to `PIVOT-LEVER`, which is translationally connected (i.e., `ALONG-ALL`) at `LOC1-LINKAGE1` (1) to the handle. The associated contact between `LOC1-LEVER` and `END2-LINKAGE2` (i.e., dart, at 2), enables the magnification of force on the dart (3). The final diagram for the toy gun illustrates the behavior associated with the spring-object, in Fig. 3.61. As with other springs represented in this section, `CG-SPRING` must be restrained from motion or compression cannot be enabled. Unlike the springs in the nutcracker, the clothes pin, and the fingernail clipper, the toy gun spring is connected at one end (`END2-SPRING`) to the barrel (1), which disables spring-object motion `ALONG-LENGTH` (2) as long as the barrel is so restrained (as mentioned in the handle

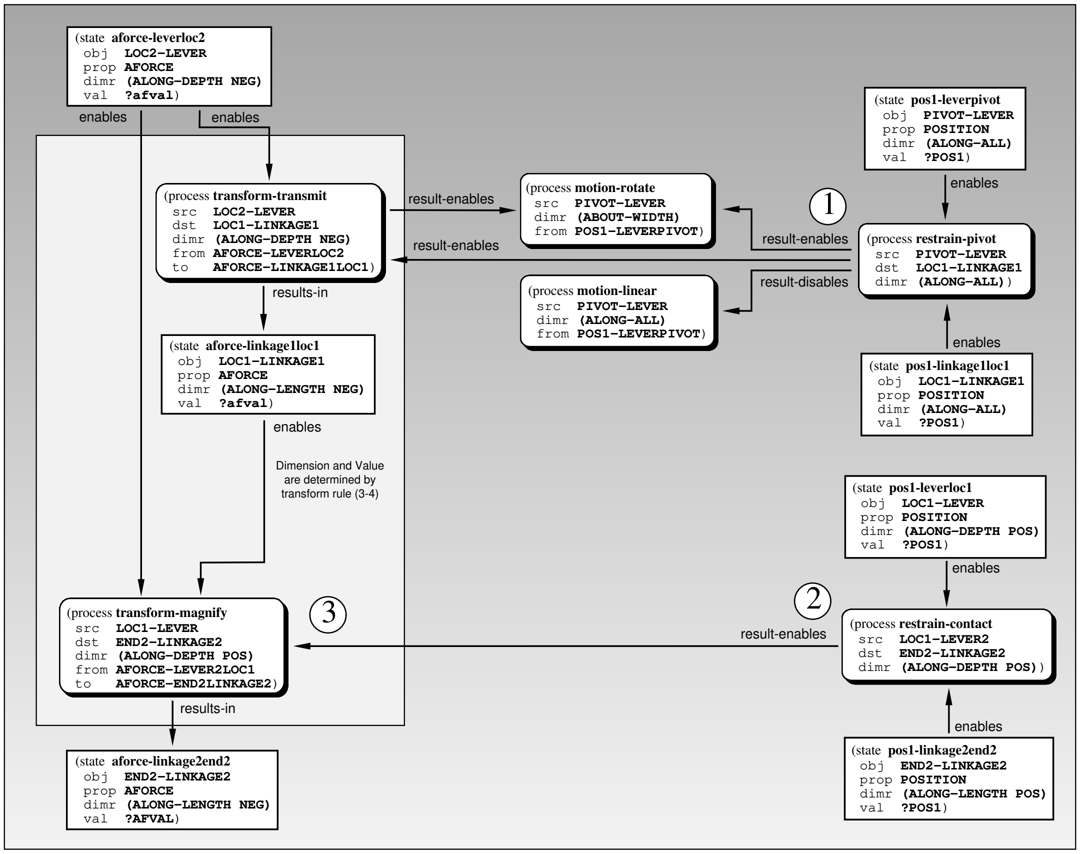


Figure 3.60 Behavioral sequence for the toy gun trigger in the arming function.

discussion for Fig. 3.58). The force applied by the dart (`END2-LINKAGE2`) to the spring-

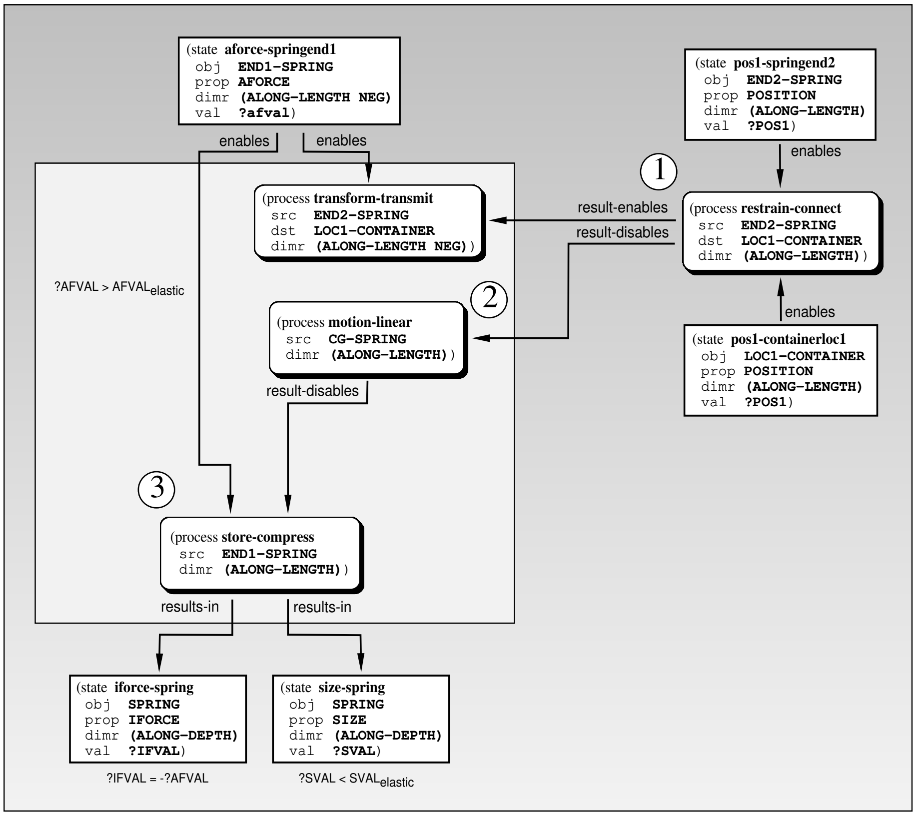


Figure 3.61 Behavioral sequence for the toy gun spring in the arming function.

object (`END1-SPRING`) thus contributes to enabling of `STORE-COMPRESS` (3) as long as the applied force exceeds the elastic threshold.