Expressing Structural Relations Among Dimension-Set Components Using the Object-Process Methodology

RECOGNITION OF GRAPHICS in technical documents involves several stages of processing and understanding. The low level is usually domain independent and includes primarily pixel-based operations, such as thinning, edge detection, and primitive symbol recognition. The intermediate level is already domain dependent. It involves matching the structural relations that, for graphics recognition, are expressed in terms of relative position of the recognized primitives with respect to each other. The structural relations stem from an underlying model, which has a specified semantics. While this model-based matching is done at the intermediate level, the semantic interpretation itself is done at the high level. The high-level interpretation, which is strictly domain dependent, can be viewed as a graphical language understanding. The language, specified by drafting standards, was originally intended solely for use by humans. However, advanced research, analogous to natural language processing, should aim at graphical language understanding. Structural relations among primitives are the relative location and orientation of one primitive with respect to the other primitives in the drawing. Description of such relations can be expressed by a number of methods, including web grammars[1] and finite automata.[2] These methods are powerful in that they capture and formalize the entire spectrum of possible primitive combinations. However, being declarative in nature, they do not prescribe a particular algorithm for checking whether certain structural relations exist. Such algorithms are derivable from *object-process diagrams* (OPDs),[3] which are the tools we employ for structural analysis of primitives. The domain taken as a case in point to demonstrate the applicability of the approach is that of mechanical engineering drawings, but OPDs can be applied to other related domains, such as utility maps, cadastral maps, electric schemes, piping diagrams, etc.

Dov Dori
Technion, Israel Institute of Technology

The object-process approach views objects and processes as complimentary things in the world. Objects are things that exist in the world (universe of interest), while processes are the things that change the state of objects. A change of state means a change in at least one value of the object's attribute, including the change of the object's attribute Existence. Such change may be either from negative to positive, i.e., generation of the object, or from positive to negative, i.e., destruction of the object. Processes are vital for describing dynamic systems and computations of soft attributes, i.e., attributes of objects that are derivable from hard attributes—the basic attributes, without which the object cannot be uniquely identified. For example, the two endpoints of a line segment are hard attributes, while its length is a soft attribute, since it can be computed from the two endpoints.

THE WIRE CONCEPT

Because engineering drawings are static in nature, the graphic symbols are related to each other through structural relations rather than through processes. Binary structural relations express long-term associations between pairs of objects. The three most prevalent structural relations are the *aggregation-particularization* (whole-part, or "part-of") relation, the *generalization-specialization* (gen-spec, or "is-a") relation, and the *characterization* relation. OPDs[3,4] provide a graphic compact and concise description of systems' structure and behavior.

Due to their frequent usage, the three structural relations are denoted by solid (black) triangles, blank (white) triangles, and solid-on-blank triangles, respectively. The characterization relation symbol is demonstrated in Figure 1, which is an OPD of the object Point.

A *simple object* is an object that, in the context of the system under consideration, has no and needs no further decomposition and/or characterization and/or specialization. A *compound object* is an object that is not simple. The name of a simple class (e.g., integer) is distinguished from that of a compound one in that it starts with a lowercase letter. Because integer is a simple object, so are X and Y, which are instances of the simple class integer. Point is a compound object, which is characterized by the two simple Point instances X and Y.

A *hard* attribute is an attribute whose value for an instance of an object class must be provided in order for the instance to be unambiguously defined. As denoted in the OPD of Figure 1, X and Y are hard attributes of Point.

A *soft* attribute is an attribute that is computed from other (hard and/or soft) attributes. The inner triangle of the soft attribute characterization relation symbol as shown in the legend of Figure 4 is blank, as opposed to the solid inner triangle symbolizing hard attribution.

While objects are denoted by rectangles, processes are denoted by ellipses. Figure 2 is an OPD that describes Point and its constructor process. The object Point is constructed by the constructor Point inscribed within an ellipse. It gets two integers as inputs and constructs an instance of Point.

Inputs to a process that are not affected by that process are denoted by an instrument link (blank circle, see Fig. 2 legend). If the input object is affected, then it should be connected to

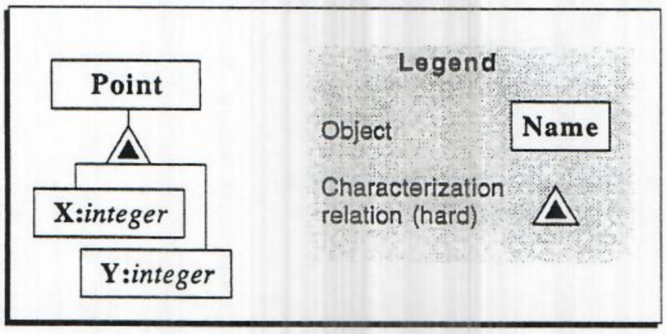

Figure 1. An OPD of the object Point.

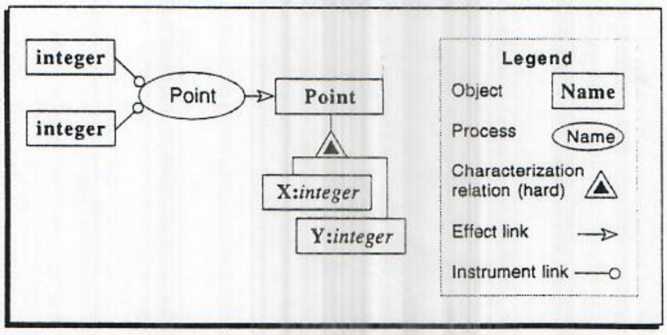

Figure 2. An OPD of the object Point and its constructor process.

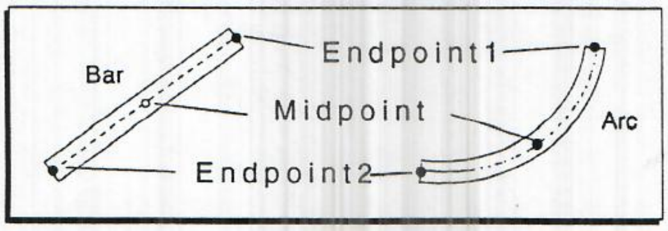

Figure 3. Bar and Arc with their endpoints and midpoints.

INFO@SIGS

SIGS Publications, Inc., 71 West 23rd Street, New York, NY 10010; 212.242.7447; Fax: 212.242.7574; email: info@sigs.com; www: http://www.sigs.com

ARTICLE SUBMISSION

To submit article proposals, outlines, and manuscripts; industry news; press briefings; or letters to the editor, please contact: Richard S. Wiener, Editor, ROAD, 135 Rugely Court, Colorado Springs, CO 80906; Phone/fax: 719.579.9616; email: rswiener@elbert.uccs.edu

CUSTOMER SERVICE

For customer service in the United States, please contact: P.O. Box 5050, Brentwood, TN 37024-5050; 800.361.1279; Fax: 615.370.4845; email: subscriptions@sigs.com
For customer service outside the United States, please contact: Subscriptions Department, Tower House, Sovereign Park, Market Harborough, Leicestershire, LE16 9EF UK; +44.(0)1858.435.302; Fax: +44.(0)1858.434.958

SIGS BOOKS

For information on any SIGS book, contact: Don Jackson, Director of Books, SIGS Books, Inc., 71 West 23rd Street, New York, NY 10010; 212.242.7447; Fax: 212.242.7574

SIGS CONFERENCES

For information on all SIGS Conferences: 212.242.7515

BACK ISSUES

To order back issues, please contact: Back Issue Order Department, SIGS Publications, 71 West 23rd Street, New York, NY 10010; 212.242.7447; Fax: 212.242.7574

LIST RENTALS

For information on list rentals, please contact: Rubin Response, 708.619.9800; Fax: 708.619.0149

REPRINTS

For information on ordering reprints, please contact: Duane Dagen, Reprint Management Services, 505 East Airport Road, Box 5363, Lancaster, PA 17601; 717.560.2001; Fax: 717.560.2063

ADVERTISING

For advertising information for any SIGS publication, please contact:

East Coast/Europe	Gary Portie
Central US	Elisa Marcus
Recruitment	Michael Peck
Exhibit Sales, West Coast:	Kristine Viksnins
Exhibit Sales, East Coast:	Sarah Olszewski

212.242.7447; Fax: 212.242.7574

West Coast	Diane Fuller

408.255.2991; Fax: 408.255.2992

SIGS HOME PAGE

To access the SIGS Home Page on the World Wide Web: http://www.sigs.com

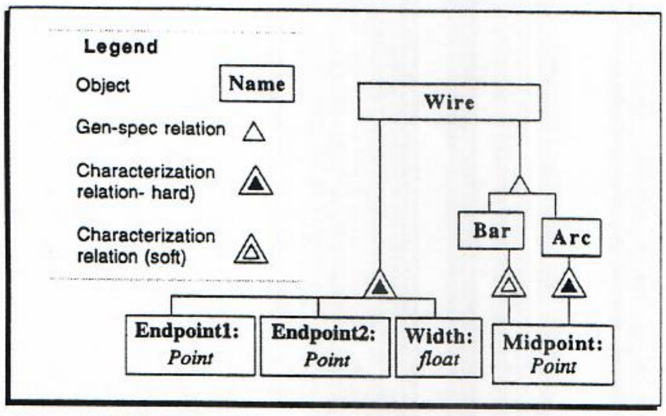

Figure 4. OPD of Wire and its specializations Bar and Arc.

the process with an effect link (arrow) rather than with an instrument link. For example, the two instrument links between the integers and the Point constructor in Figure 2 denote the fact that the integers are required for the process, but they are not affected by it.

Bar is a straight line segment with nonzero width. An example of a bar and an arc is shown in Figure 3. Bar has three hard attributes: Width and two endpoints, Endpoint1 and Endpoint2.

The three attributes that uniquely represent Bar, however, are not sufficient for Arc because an infinite number of arcs can be constructed without an additional constraint. Therefore, an additional attribute must be assigned to Arc. There are a number of options, including radius, center, and angle. However, no direction (clockwise and counterclockwise) is imposed on the direction from Endpoint1 to Endpoint2, for each one of these three possible attributes we need an extra attribute. For radius and center we need to specify the direction, and for angle we need to specify on what side of the chord connecting the two endpoints the arc lies. Besides, in actual engineering drawings the radius, center, or angle may not be readily available, but what should always be available is a third point on the arc. Therefore, Midpoint is selected as the

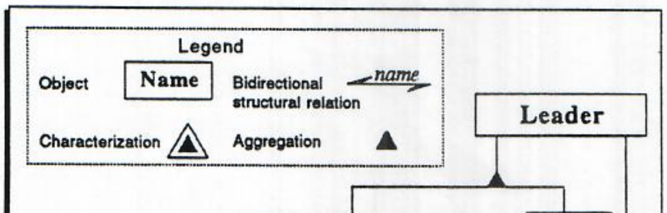

Figure 5. The structural relations cowire and orthowire among Leader's components Arrowhead, Tail, and Reference.

fourth attribute of Arc. The perpendicular bisector tracing algorithm[5] uses this point to detect arcs.

Midpoint is defined as the point lying midway between the two Bar endpoints. Arc is a specialization of Bar. For Arc, Midpoint is a hard attribute, because it cannot be computed from the two endpoints as in Bar, and there is an infinite number of arcs with the same width and different Midpoints. Hence, Midpoint is a soft attribute for Bar and a hard attribute for Arc. This is denoted in Figure 2 by drawing the actual midpoint filled (black) for Arc and blank (white) for Bar.

Figure 4 is an OPD of Wire and its specializations Bar and Arc. Bar inherits from Wire the three hard attributes Endpoint1, Endpoint2, and Width. Arc, in addition, inherits the hard attribute Midpoint. For Bar, Midpoint is a soft attribute, as discussed below. The syntax for the three constructions is

```
Wire(Endpoint1, Endpoint2, Width)
Bar(Endpoint1, Endpoint2, Width)
Arc(Endpoint1, Endpoint2, Width, Midpoint)
```

The default Width value is zero. This enables fast construction of Euclidean straight line segments and arcs, which are needed frequently for various purposes. Thus, Bar(Endpoint1, Endpoint2) and Arc(Endpoint1, Endpoint2, Midpoint) construct zero-width Bar and zero-width Arc, respectively.

To reduce the diagram complexity, the details of Point shown in Figure 1 are omitted from Figure 4 for all three points. This diagram simplification strategy is applied throughout the set of OPDs. The complete information can be obtained by looking at all the relevant OPDs rather than just a single OPD.

The importance of generalizing bars and arcs to wires is in the generation of an abstract object class—Wire—which enables us to think and express structural relations at a higher level of abstraction. Thus, for example, we can talk about the cowiring relation of two wires, which is a short way of saying "colinearity if the two wires are bars, co-circularity with the same radius if the two wires are arcs, and tangency if one wire is an arc and the other is a bar."

ARROWHEADS AND LEADERS

Arrowheads are key objects for dimensioning recognition. We refer to arrowheads whose shape is a solid isosceles triangle because they conform with both ISO and ANSI standards and the vast majority of drawings use them. As shown in Figure 5, Arrowhead is a Bar whose hard attributes are Tip and Back, which are Bar's Endpoint1 and Endpoint2, respectively, and Width, which is the length of the triangle base.

Leader is an aggregation of (consists of) an Arrowhead and a Tail, which is a specialization of Wire. Because Wire is a generalization of Arc and Bar, we have two specializations of Leader: one consisting of an Arrowhead and a Bar (a Linear Leader) and the other consisting of an Arrowhead and an Arc (an Angular Leader). Using the OPD in Figure 5, the graphic-symbolic representation of these objects and the structural relations among them, from which these assertions can be inferred, is again very compact and concise.

Structural binary relations are relations between two objects that hold in the system regardless of the point in time when the observation is made. Common examples for such relations are aggregation, generalization, and characterization, which have special symbols (black, white, and black-on-white triangles, respectively). Structural relations other than these three prevalent ones are denoted in an OPD by the relation name written along the line ending with one or two arrows that leads from one object to another. An example is the relation "cowire" between Arrowhead and Tail in Figure 5.

Cowiring is a generic term whose geometric interpretation depends on the specialization of Wire. The meaning of cowiring is colinearity if both wires are Bars, concentricity if both wires are Arcs, and tangency if one wire is a Bar and the other is an Arc. Thus, if a Leader has an Arc Tail, for example, that Arc should be tangent to the Bar representing the Arrowhead. Cowiring has three optional parameters: gap, tolerance, and knee, all with default values of zero. Gap is the distance between the two cowiring wires, tolerance is the amount of deviation from this gap, and knee is the maximal angle of deviation from 180°. The method Wire::cowire(Wire1, Wire2, Gap, Tolerance, Knee) is a Boolean function that returns True if Wire1 cowires with Wire2, and False otherwise. This method can be used to detect dashed wires as well as verify the existence of a Leader.

The method Wire::conwire(Wire1, Wire2, Gap, Tolerance, Knee) is a Boolean function that specializes Wire::cowire() in that it returns True only if, in addition to the requirements of the general method, it requires also that Wire1 be of the same specialization as Wire2, i.e., either both are Bars or both are Arcs,

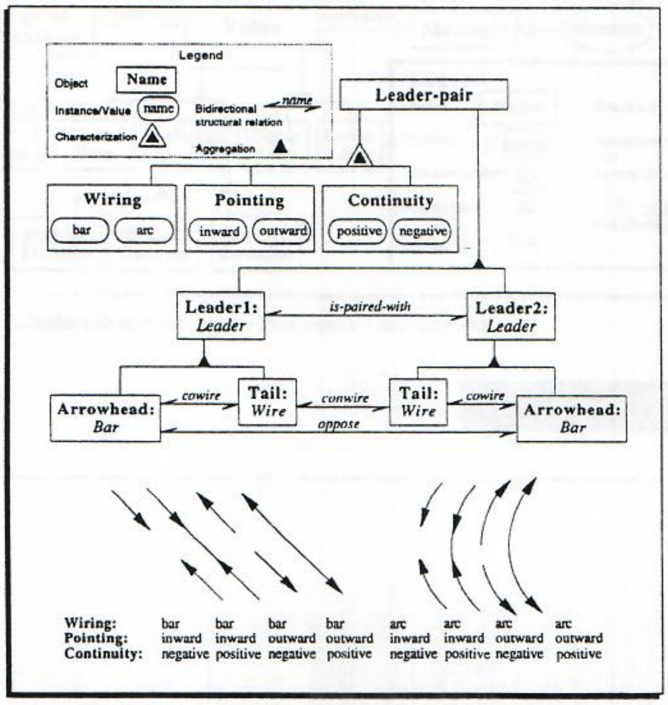

Figure 6. The structural relations cowire and oppose among Leader-pair's components and the eight Leader-pair combinations.

and False otherwise. This method can be used to detect dashed wires and, as shown below, pair Leaders into Leader-pairs.

Orthowiring is another generic term whose geometric interpretation depends on the specialization of Wire. The meaning of orthowiring is perpendicularity if both wires are Bars, perpendicularity of the Bar to the tangent of the Arc at the point of intersection if one wire is an Arc, and perpendicularity to the tangents of the two Arcs at the point of intersection if both wires are Arcs. As shown in Figure 5, Arrowhead, as a Bar, has to be orthowire to the Reference, which is a Wire, to which the Arrowhead points.

Similar to cowiring, orthowiring has three optional parameters: gap, tolerance, and knee, all defaulting to zero. Gap is the minimal distance between one wire (Reference in our case) and an edge of the other wire (the Tip of the Arrowhead in our case), tolerance is the amount of deviation from this gap, and knee is the maximal angle of deviation from 90°. The method Wire::orthowire(Wire1, Wire2, Gap, Tolerance, Knee) is a Boolean function that returns True if Wire1 is orthowire to Wire2, and False otherwise. This method can be used to detect References as well as verify the existence of a Leader.

LEADER-PAIRS

A Leader-pair is a pair of Leaders, Leader1, and Leader2, that satisfy the structural relation "is-paired-with," shown in the OPD of Figure 6. As the OPD shows, for two Leaders to be paired, two