Synthesis of Solid Models of Polyhedra from their Orthogonal Views using Logical Representations

Gabriela Garza
Instituto de Investigaciones Eléctricas
Unidad de Sistemas Informáticos
AP 1-475, Cuernavaca, Mor., México
Tel. 18-38-11, ext. 7168
Fax 18-26-58
ggarza@sgi.iie.org.mx

Luis Pineda
Instituto de Investigaciones Eléctricas
Unidad de Sistemas Informáticos
AP 1-475, Cuernavaca, Mor., México
Tel. 18-38-11, ext. 7008
Fax 18-26-58
luis@sgi.iie.org.mx

Abstract

In this paper a representational language of a logical kind which is expressive enough to model concepts of descriptive geometry is presented. The language is employed to produce solid models of polyhedra from representations of their orthogonal projections. The synthetic process has as its input well-formed expressions representing the geometrical entities and relations of bidimensional orthogonal views and employs expressions representing drafting concepts of descriptive geometry to produce expressions denoting the 3D objects contituting the final polyhedra. The production of 3D models of solid objects from their orthogonal views has a significant history within computer graphics and CAD. Most previous approaches are based on numerical algorithms modeling geometrical and topological constraints of the problem domain in a quantitative fashion. However, in this approach, geometrical objects are referred to through expressions of a declarative language, and the final object is produced through symbolic inference.

1. Introduction

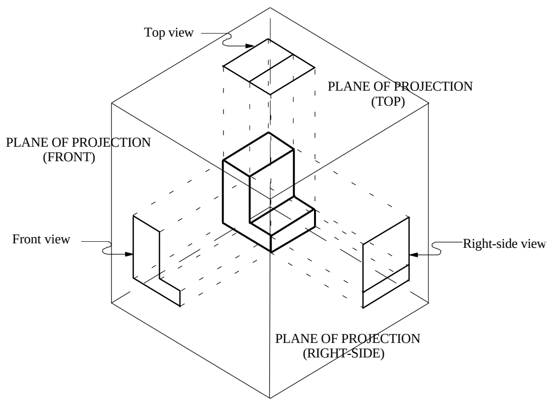
The production of solid models of polyhedra from orthogonal views has been the subject of a large number of studies. Research addressing this problem can be traced as far back as Idesawa's 1973 study [Idesawa73]. Other systems have been produced by Lafue [Lafue73], Sakurai and Gossard [Sakurai83], Haralick and Queeney [Haralick82], Preiss [Preiss84]. A good survey of all of these algorithms is presented by Nagendra and Gujar [Nagendra88]. Most of these systems employed a bottom-up approach which starts from the identification of the basic 2D points for the sequential production of 3D points, lines, surfaces to arrive to the final 3D model. Although orthogonal views are normally unambiguous, synthetic procedures based on the computation of local properties of geometric entities generate false 3D elements that have to be removed for producing a valid configuration. Validity is tested in these approaches through mathematical criteria. Constraints at this level of abstraction are, for instance, that a valid polyhedra have to satisfy Euler's rule. Another important characteristic of these methods is that the information concerning the coordinate positions of dots and lines in orthogonal views is especified in three different 2D coordinated systems, one for each view. For instance, the top view corresponds to the x-y plane, the front view to the x-z plane and the right view to the y-z plane. 3D coordinate positions are computed from these basic 2D spaces.

In normal drafting tasks, on the other hand, people apply more abstract interpretative concepts relating views, faces, projections and constraints on these notions. For instance, two adjacent faces must not meet on the same plane and an edge must be shared by at most two faces. Rather than starting the interpretation process by looking at individual dots, people consider polygons in especific views as projection of faces during the whole of the interpretation process. This view suggests a top-down interpretation process for the construction of solid models. In addition, it is considered that from the point of view of people all views are represented in a single piece of paper, which is a 2D space. The axes determining the position of the views relative to each other are drawn relative to a 2D reference space for the drawing as a whole, and a number of auxiliary geometrical elements commonly used in drafting practices, like construction lines, are defined relative to this space too. This qualitative description of drawings through a formal declarative language follows the lines of the graphical language developed for the GRAFLOG system [Pineda89,92], which allows us to make systematic complex reference to graphical entities and relations appearing in orthogonal drawings. However, while the GRAFLOG system is only concerned with the production of 2D drawings, the aim of the present theory is to obtain solid models. The purpose of this work is to express explicitly the knowledge required for the interpretation of orthogonal projections and the production of isometric views at the knowledge level.

In Section 2 a characterization of the kind of polyhedra that can be interpreted by the system is presented and some examples of the graphical concepts that have to be model to support the interpretation task are illustrated. Then, an intuitive introduction to the expressions of the representational language required for referring to individuals and relations involved in drafting concepts is given. In Section 3 a formal definition of the syntax and semantics of the representational language is presented. The language is essentially a many-sorted first-order logical language with equality, function symbols and lambda abstraction. In Section 4 a set of concepts of descriptive geometry required for the interpretation of orthogonal views is presented. Finally, in Section 5 the heuristic procedure that uses these expressions to synthesize the solid model is illustrated with the help of an example.

2. Modeling Graphical Concepts of Descriptive Geometry

Consider Figure 1a in which the orthogonal projections or views of a polyhedron are shown. As can be seen the 3D representation of the solid, enclosed in a transparent box, is decomposed in a number of 2D representations, usually three (namely, top, front and right-side views). In normal technical drafting practices people are given these three views in a planar representation relative to a 2D coordinate system as shown in Figure 1b, and his or her task is to produce a 3D representation (isometric, oblique or perspective). Although these drawing are interpreted as 3D objects, they are not really tridimensional as they are drawn in a piece of paper, and the reasoning process applied by people for the construction of isometric views acts upon bidimensional representations of objects and produces tridimensional representations of the sought polyhedra (presumably).


Top view
PLANE OF PROJECTION (TOP)
PLANE OF PROJECTION (FRONT)
Front view
Right-side view
PLANE OF PROJECTION (RIGHT-SIDE)
(a)

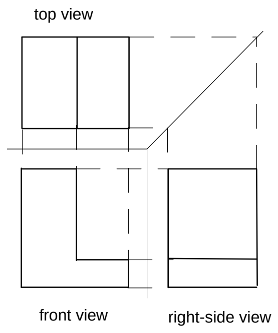


Figure 1 Orthogonal projections or views of a polyhedron.

It is worth highlighting that this notion of reasoning in 2D considers not only the 2D objects appearing explicitly on the drawing, but also other objects, like constructions lines, employed to relate objects in different views. In particular, a graphical reasoning task in this problem-solving domain consists in verifying whether objects in different views correspond to the projections of a 3D object. Whenever such a relation holds it is possible to postulate that a certain 3D object is being represented through its orthogonal views.

Next this notion of graphical reasoning is illustrated with the help of a simple example. When a surface is perpendicular to a plane, the projection of the surface on to that plane is a line; otherwise the projection is a polygon, as can be seen in Figure 1a in which the top surface of the solid is projected as a line in the front and right-side views, and as a polygon in the top view. A typical situation of a surface that is perpendicular to the top plane of projection and inclined to the others is illustrated in Figure 2. In order that people can indentify the line and two polygons of Figure 2, when they occur in the context of a meaningful and complete drawing as projections of the same surface, a number of conditions must hold that must be verified through a reasoning process. In technical drafting practices the conditions are identified with the help of a number of construction lines. In our example, these conditions are as follows:

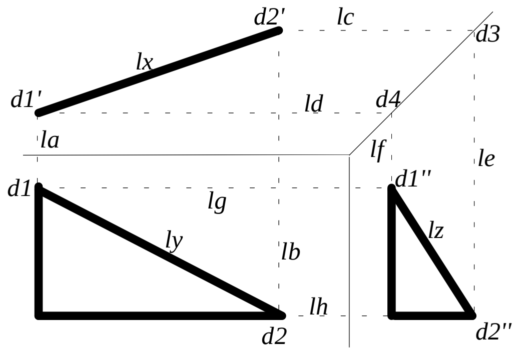


Figure 2 Orthogonal views of a surface perpendicular to the top plane and inclined to the others.

1. Relation between the line in the top view and the polygon in the front view:

1.1. A vertical construction line *la* passing through the left dot of the line in top view must intersect the left-most dot of the polygon in the front view.

1.2. A vertical construction line *lb* passing through the right dot of the line in the top view must intersect the right-most dot of the polygon in front view.

2. Relation between the polygon in the front view and the polygon in the right-side view:

2.1. A horizontal construction line *lg* passing through the top-most dot of the polygon in the front view must intersect the top-most dot of the polygon in the right-side view.

2.2. A horizontal construction line *lh* passing through the bottom dot of the polygon in the front view must intersect the bottom dot of the polygon in the right-side view.

3. Relation between the line in the top view and the polygon in the right-side view:

3.1. A vertical construction line *le* passing through the intersection between the 45° axis and the horizontal construction line *lc* -dot *d3*- intersects the right-most dot of the polygon in the right-side view. The line *lc* passes through the top-most dot of the line in the top view and intersects *d3*.

3.2. A vertical construction line *lf* passing through the intersection between the 45° axis and the horizontal construction line *ld* -dot *d4*- intersects the left-most dot of the polygon in the right-side view. The line *ld* passed through the bottom dot of the line in the top view and intersects *d4*.

4. Relation between the lines in the front view with the lines in the right-side view:

A line of the polygon in the front view corresponds to a line of the polygon in the right-side view if a common reference can be identified through the top view. For instance, consider the line *ly* in the front view and its corresponding line *lz* in the right-side view. The line *lz* is defined by dots *d1''* and *d2''*, where *d1''* is defined by the intersection of the horizontal contruction line *lg* passing through *d1* in the front view, and the vertical construction line *lf* passing through *d4* which in turn is determined by the intersection between the 45° axis and the horizontal construction line *ld*. The line *ld* passes through *d1'* which is defined by the intersection of *lx* and the vertical construction line *la*, which passes through dot *d1*. The dot *d2''* defining *lz* is obtained as above but considering the extreme dot *d2* of *ly*. Relations between other lines of the front and right-side view can be identified by similar constructions.

For verifying whether these conditions hold, it is required to have a representational language expressive enough to state such geometrical descriptions. Note that natural language descriptions 1 to 4 involve relative clauses, conjunctions, disjunctions, reference to individuals and groups of individuals, etc. It is not clear how these descriptions can be expressed dynamically (through an interactive dialog with the final user) in traditional languages of the kind commonly used in graphics and CAD systems, like for instance, object oriented programming languages. In the current approach a many sorted first-order logical language augmented with equality and lambda abstraction is employed. Through this language complex descriptions can be stated and their denotations can be obtained through the interpretation process. The synthetic program that produces the solid object has as its input a collection of expressions representing the orthogonal views and uses the language interpreter to compute the actual reference of expressions needed in the evaluation of constraints of the task and in the production of the solid model. In our representational language (presented below in Section 3), the expression denoting the objects and relations illustrated in Figure 2 is as follows:

$$
\begin{array}{l}
\lambda x_{line}, y_{polygon}, z_{polygon} \\
\quad (\mathit{dot\_on\_line}(\mathit{p\_left}(y_{polygon}), \mathit{proj}(\mathit{top}, \mathit{front}, \mathit{left}(x_{line}))) \;\wedge \\
\quad (\mathit{dot\_on\_line}(\mathit{p\_right}(y_{polygon}), \mathit{proj}(\mathit{top}, \mathit{front}, \mathit{right}(x_{line}))) \wedge \\
\quad (\mathit{dot\_on\_line}(\mathit{p\_left}(z_{polygon}), \mathit{proj}(\mathit{top}, \mathit{right}, \mathit{down}(x_{line}))) \wedge \\
\quad (\mathit{dot\_on\_line}(\mathit{p\_right}(z_{polygon}), \mathit{proj}(\mathit{top}, \mathit{right}, \mathit{up}(x_{line}))) \wedge \\
\quad (\mathit{dot\_on\_line}(\mathit{p\_up}(z_{polygon}), \mathit{proj}(\mathit{front}, \mathit{right}, \mathit{up}(x_{line}))) \wedge \\
\quad (\mathit{dot\_on\_line}(\mathit{p\_down}(z_{polygon}), \mathit{proj}(\mathit{front}, \mathit{right}, \mathit{down}(x_{line}))) \wedge \\
\quad \forall w_{line}\, (\mathit{line\_of\_polygon}(w_{line}, y_{polygon}) \rightarrow \\
\qquad\qquad \mathit{line\_of\_polygon}(\mathit{line}(\mathit{intersection}(\mathit{proj}(\mathit{tv}, \mathit{rv}, \mathit{intersection}(\mathit{proj}(\mathit{fv}, \mathit{tv}, \mathit{left}(w_{lline})), x_{line})), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{proj}(\mathit{fv}, \mathit{rv}, \mathit{left}(w_{line}))), \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{intersection}(\mathit{proj}((\mathit{tv}, \mathit{rv}, \mathit{intersection}(\mathit{proj}(\mathit{fv}, \mathit{tv}, \mathit{right}(w_{line})), x_{line})), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{proj}(\mathit{fv}, \mathit{rv}, \mathit{right}(w_{line})))), \\
\qquad\qquad\qquad\qquad z_{polygon}))
\end{array}
$$

Expression 1 Expression of the logical languaje denoting the objects and relations illustrated in Figure 2.

As can be seen, expression 1 denotes a function of sort *line* x *polygon* x *polygon* → *bool* which takes the line in the top view and the polygons in the front and right-side views, and returns the value *true* if the line and the two polygons correspond to the projections of a surface of a solid. The operator symbol *dot_on_line* in the body of expression 1 is interpreted as a function that takes a dot and a line and produces as its value *true* if the dot is on the line and *false* otherwise. The symbol *proj* denotes a function which takes as its arguments two views and a dot and has as its value a line, which is the construction line, vertical or horizontal, that passes through the dot in the first view and relates the two views. For instance, the construction line through the point *d1'* that relates the top and front views is expressed in the language as *proj*(*top*,*front*,*d1'*), and the construction line passing through *d1'* that relates the top and right-side views is expressed by *proj*(*top*,*right*,*d1'*) and denotes the line *lf*. The expression *left*(*l*) denotes the left-most dot of line *l* and, on a similar way, the symbols *right*, *up*, *down*. The interpretations of the operator symbols *p_left*, *p_right*, *p_up*, *p_down* are the left-most, right-most, etc., dot of a polygon. The language makes explicit use of quantifiers and variables of the different sorts, and employs also logical constant symbols of first order logic. Although expressions are formally defined in a prefix form, we use infix notation when appropiated for clarity. The development of the syntax and semantics of representational languages and its interpreter is one main focus of this work, and in Section 3 a full specification of this language is presented.

Consider now that the situation depicted in Figure 2 never occurs out of contex, but it rather appears embedded within a whole orthogonal configuration with many graphical symbols in which the projections of a given entity in one view into the others cannot be known by simple inspection of the drawing in most situations. Consider Figure 1b in which several lines and polygons can be discerned in each of the views. How do we know that the right rectangle of the top view corresponds to a line on each to the other two views? The first approximation to the problem would be to consider an exhaustive search in which every individual entity (dots, lines and polygons) of one of the views is compared with all objects in the other views. This approach is, however, not practical because the exponential nature of the search process. In normal human drafting practices people do not do this kind of search and some heuristics must be involved in the selection of the entities that are likely to match. One strategy for solving the problem would be to order the views and scan objects in views according to that order. Polygons in views can also be ordered for the scaning process and the search can be performed in a way that only relevant paths are explored. This process needs to be guided with the help of a number of heuristics. Our procedure will be explained in Section 5. For the present purpose what needs to be emphasized is that the final object must satisfy a number of constraints in order to be considered a valid polyhedron. Constrains can be checked in local steps of the generation process and also at a global level when the final object is produced. Global constraints on objects produced by a synthetic procedure are as follows:

i) A polyhedron must be a solid limited by a set of polygons, interpreted as faces, where each line -an edge- must be shared exactly by two faces (2-manifold polyhedron). In Figure 3a an invalid polyhedron is shown[1].
ii) Two adjacent faces of a polyhedron must not lie in the same plane (as shown in Figure 3b).
iii) Each face of a polyhedron is defined by exactly one polygon. This constraint eliminates polyhedra with holes because a surface with a hole is defined by two polygons where one of them limits the face and the other the hole. An example of this case is presented in Figure 3c. Another invalid case for this condition is shown in Figure 3d.

According to the above definition some examples of valid and invalid polyhedra are shown in figure 3.

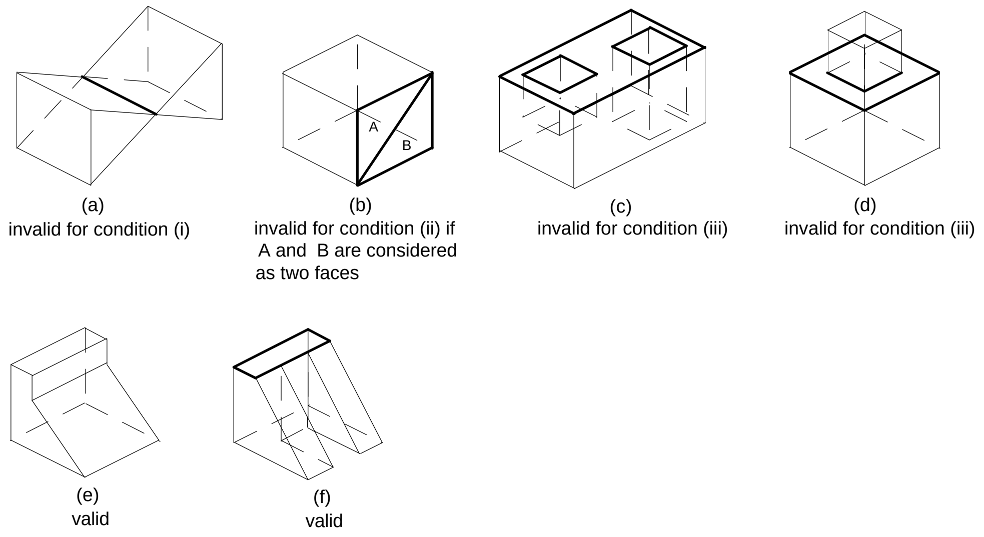

A
B
(a)
invalid for condition (i)
(b)
invalid for condition (ii) if
A and B are considered
as two faces
(c)
invalid for condition (iii)
(d)
invalid for condition (iii)
(e)
valid
(f)
valid

Figure 3 VALID AND INVALID POLYHEDRA. (a), (b), (c) and (d) are invalid polyhedra. (e) and (f) are valid polyhedra. The marked face in polyhedron (f) is considered with six sides according to condition (i).

3. Definition of a Representational Language

In this section a formal presentation of a general framework for the definition of the kind of languages that are used in this work is is presented. A language is a set of symbols and a set of rules to construct well-formed expressions. We refer to the expressions of the language that denote individuals of different geometrical kinds as *graphical terms*, or simply *terms*. The interpretation of a composite expressions is computed according to Frege's *Principle of Compositionality* [Dowty81]. This principle states that the interpretation of a composite expression depends on the interpretations of its constituting parts and their

[1]Thick lines represent the edges referred to by the condition.

mode of grammatical combination. The interpretation of a language is relative to a formal model in the logical sense. In order to define a language it is required to specify its *syntax* and *semantics*.

3.1 Syntax of a Language

The syntax of a language determines the structure of expressions. The syntactic elements of a language are:

A. A set of *sorts*,
B. A set of *basic expressions* or *symbols* for each sort.
C. A set of *syntactic rules* to construct well-formed expressions of the language.

We define the syntax of a language with a notation based on a combination of those in [Goguen78] and [Dowty81], augmented with graphical considerations as shown in [Pineda89]. We distinguish basic and composite sorts. A composite sort is formed by others sorts and has the form $s_1\ s_2 \ldots s_n ,s$ where $s_1, s_2, \ldots, s_n$, s are sorts. The interpretation of an expression of a basic sort s is an object of sort s. For instance, the interpretation of the symbol "3" of sort *integer* is the integer number -the abstract object- 3. The interpretation of an expression of a composite sort $s_1\ s_2 \ldots s_n,s$ is a function of sort $A_{s1} \times A_{s2} \times \ldots \times A_{sn} \rightarrow A_s$, where A_i is the set of all objects of sort i. For instance, the interpretation of the operator symbol *intersection* of sort *line line,dot* is a function with domain in $A_{line} \times A_{line}$ and range in A_{dot} (i.e. the intersection of two lines is the dot in which the intersection takes place).

Let S be the set of **basic sorts** and T the whole set of sorts.

The construction rules for the formation of sorts are as follows:
1) If $s \in S$ then $s \in T$.
2) If $s_1, s_2, \ldots, s_n \in T$ and $s \in T$ then $s_1\ s_2 \ldots s_n,s \in T$.

The **basic expressions** are the constant and variable symbols of the language:
1) For each sort $s \in T$, the set of constants of sort s is C_s.
2) For each sort $s \in S$, the set of variables of sort s is V_s.

Note that the language have basic constants for every sort whether it is basic or complex, but only variables of basic sorts are allowed, keeping the language as a first order one.

The **syntactic rules** are as follows (the set of well-formed expressions of sort s is E_s):
1. If $\alpha \in C_s$, then $\alpha \in E_s$.
2. If $\mu \in V_s$, then $\mu \in E_s$.
3. If $\tau_1, \ldots, \tau_n$ are elements of $E_{s1}, \ldots, E_{sn}$, respectively, and $\phi \in C_{s1\ldots sn,s}$, then $\phi(\tau_1, \ldots, \tau_n) \in E_s$.
4. If $\mu_1, \ldots, \mu_n$ are elements of $V_{s1}, \ldots, V_{sn}$, respectively, and $\alpha \in E_s$, then $\lambda\ \mu_1, \ldots, \mu_n\ \alpha \in E_{s1\ldots sn.,s}$.
5. If $\lambda\ \mu_1, \ldots, \mu_n\ \alpha \in E_{s1\ldots sn,s}$ and $\tau_1, \ldots, \tau_n$ are elements of $E_{s1}, \ldots, E_{sn}$, respectively, then $\lambda\ \mu_1, \ldots, \mu_n\ \alpha(\tau_1, \ldots, \tau_n) \in E_s$.
6. If $\mu \in V_s$ and $\beta \in E_{bool}$ then $\exists\mu(\beta) \in E_{bool}$.
7. If $\mu \in V_s$ and $\beta \in E_{bool}$ then $\forall\mu(\beta) \in E_{bool}$.

3.2 Semantics of a Language

The semantics of a language defines the interpretation of the well-formed expressions. In order to define the semantics we specify the following:
A. The *model*. A model is an ordered pair $\langle A,F \rangle$ where A (the *domain*) is a non-empty set of individuals, and F is a set of functions which assign interpretations to all constant symbols of every sort.
B. The *semantic rules* determine the interpretation of a well-formed expression.

As functions in F assign a denotation to every constant, an assignment function g is introduced to assign values to the variables. For this reason the intepretation of an expression of the language is relative to a model M and an assignment function g.

Let the **model M** be defined as follows:

Let the domain of individuals be $A = A_{s1} \cup A_{s2} \cup \ldots \cup A_{sn}$, for all $s_i \in S$.

Let the set of functions F be $\{F_{s1}, \ldots, F_{sn}\}$ for all $s_i \in S$, where F_{si} assigns an interpretation to every element of C_{si}.

The **semantic rules** are the following (following [Dowty81] we use the notational convention $[[\alpha]]^{M,g}$ for representing the interpretation or denotation of an expression α relative to a model M and function g; a function $g^{v/k}$ is the same as g but assigns the value k to the variable v):

1. If $\alpha \in C_s$, then $[[\alpha]]^M = F_s(\alpha)$.

2. If $\mu \in V_s$, then $[[\mu]]^{M,g} = g(\mu)$.

3. If $\tau_1, \ldots, \tau_n$ are elements of $E_{s1}, \ldots, E_{sn}$, respectively, and $\phi \in E_{s1\ldots sn,s}$, then $[[\phi(\tau_1, \ldots, \tau_n)]]^{M,g} = [[\phi]]^{M,g} ([[\tau_1]]^{M,g}, \ldots, [[\tau_n]]^{M,g})$.

4. If $\mu_1, \ldots, \mu_n$ are elements of $V_{s1}, \ldots, V_{sn}$, respectively, and $\alpha \in E_s$, then $[[\lambda\, \mu_1, \ldots, \mu_n\, \alpha\,]]^{M,g}$ is that function h from $A_{s1} \times A_{s2} \times \ldots \times A_{sn}$ into A_s such that for all objects $k_1, k_2, \ldots, k_n$ in $A_{s1}, A_{s2}, \ldots, A_{sn}$, respectively, $h(k_1, k_2, \ldots, k_n)$ is equal to $[[\alpha]]^{M,g\ \mu 1/k1,\ldots,\ \mu n/kn}$.

5. If $\lambda\, \mu_1, \ldots, \mu_n\ \alpha \in E_{s1\ldots sn.,s}$ and $\tau_1, \ldots, \tau_n$ are elements of $E_{s1}, \ldots, E_{sn}$, respectively, then $[[\lambda\, \mu_1, \ldots, \mu_n\ \alpha\, (\tau_1, \ldots, \tau_n)]]^{M,g} = [[\alpha(\mu_1/\tau_1, \ldots, \mu_n/\tau_n)]]^{M,g}$.

6. If $\mu \in V_s$ and $\beta \in E_{bool}$ then $[[\exists\mu(\beta)]]^{M,g} = \mathbb{T}$ iff for some value assignment g'such that g' is exactly like g except possibly for the individual assigned to μ by g', $[[\beta]]^{M,g'} = \mathbb{T}$.

7. If $\mu \in V_s$ and $\beta \in E_{bool}$ then $[[\forall\mu(\beta)]]^{M,g} = \mathbb{T}$ iff for every value assignment g'such that g' is exactly like g except possibly for the individual assigned to μ by g', $[[\beta]]^{M,g'} = \mathbb{T}$.

The **semanctic rule of interpretation relative to a model *M*** is the following:

If $\alpha \in E_s$, then $[[\alpha]]^M = a_s$, where $a_s \in A_s$, if $[[\alpha]]^{M,g} = a_s$ for every value assignment g.

3.3 Syntactic and Semantic Definition of Language *L*

Now the definition of language L for the representation of drawings is presented. A particular interpretation domain A for a specific model M contains all graphical symbols that can occur in orthogonal views. (In the following text, we use a notational convention by which individuals of the domain are written in normal text and constant symbols of the object language in italics).

Let S be the set of basic sorts as follows:

S = {*bool, view, integer, real, dot, line, iline, path, polygon, dot3d, line3d, path3d, polygon3d*}

where

- *bool* represents the sort *boolean*
- *view* represents the sort *orthogonal view*
- *integer* representes the sort *integer number*
- *real* represents the sort *real number*
- *dot* represents the sort *2D dot*
- *line* represents the sort *2D line segment*
- *iline* represents the sort *2D infinite line*
- *path* represents the sort *2D path*
- *polygon* represents the sort *2D polygon*
- *dot3d* represents the sort *3D dot*
- *line3d* represents the sort *3D line segment*

- *path3d* represents the sort *3D path*
- *polygon3d* represents the sort *3D polygon*

Let V be a denumerable set of variables of all basic sorts in S.

Let A be the set of individuals formed by the union of the following sets:

A_{bool} = $\{\mathbb{T}, \mathbb{F}\}$ is the set of boolean values.
A_{view} = {top, front, right }
A_{real} = {r | r is a real number}.
$A_{integer}$ = {r | r is an integer number}.
A_{dot} = $\{d_1, d_2, \ldots , d_n\}$.
A_{line} = $\{l_1, l_2, \ldots, l_n\}$.
A_{iline} = $\{i_1, i_2, \ldots, l_n\}$.
A_{path} = $\{\text{empty}, k_1, k_2, \ldots, k_n\}$.
$A_{polygon}$ = $\{p_1, p_2, \ldots, p_n \}$.
A_{dot3d}, A_{line3d}, A_{path3d} , $A_{polygon3d}$ are finite sets of the corresponding kinds of objects.

Every set A_s has an "error" element such that $\text{error}_{\text{dot}} \in A_{dot}$, $\text{error}_{\text{line}} \in A_{line}$, etc. Similarly, every set C_s has a symbol that denotes the error element, for example, the symbol $error_{dot}$ is an element of C_{dot} and $F_{dot}(error_{dot}) = \text{error}_{\text{dot}}$. Errors occur when the interpretations of expressions are partial functions [Pineda89,92].

Now we present some of the constant symbols of the language L and their denotations. Let the set of basic constants C be formed by the union of the following sets, and their interpretations as follows (for simplicity, we refer to function F_s for some s only as F):

- The set C_{bool} = $\{ \mathcal{F}, \mathcal{T}\}$ such that $F(\mathcal{F}) = \mathbb{F}$ y $F(\mathcal{T}) = \mathbb{T}$.
- The set C_{view} = {*top*, *front*, *right*} such that F(*top*)=top, F(*front*)=front and F(*right*)=right.
- The infinite set of real numerals C_{real} = $\{r_1, r_2, \ldots \}$, such that $F(r)$=real number r.
- The infite set of integer numerals $C_{integer}$ = $\{n_1, n_2, \ldots\}$, such that $F(n)$=integer number n.
- The set C_{dot} = $\{d_1, d_2, \ldots ,d_n\}$ such that $F(d_i)=\text{d}_\text{i}$.
- The set C_{line} = $\{l_1, l_2, \ldots ,l_m\}$ such that $F(l_i)=\text{l}_\text{i}$.
- The set C_{iline} = {*verAxis, horAxis, axis45*}: the axes of orthogonal drawings..
- The set C_{path} = $\{empty, k_1, k_2, \ldots, k_n\}$ such that $F(empty)$=empty, $F(k_i)=\text{k}_\text{i}$.
- The set $C_{polygon}$ = $\{p_1, p_2, \ldots, p_t\}$ such that $F(p_i)=\text{p}_\text{i}$.
- Sets C_{dot3d}, C_{line3d}, C_{path3d}, $C_{polygon3d}$, are defined in a similar way as C_{dot}, etc.
- The set $C_{bool\ bool,\ bool} = \{\wedge, \vee, \Rightarrow\}$ of symbols whose denotations are the logical operations *and*, *or* and *implication*, respectively.
- The set $C_{bool,\ bool}$ = $\{\neg\}$, where the denotation of symbol $\neg$ is the logical operation *not*.
- The set $C_{real,real}$ = {*rad*} where the denotation of symbol *rad* is the function whose value is the equivalence in radians of a given angle in degrees.
- The set $C_{real\ real,\ dot}$ = {*dot*}, where symbol *dot* denotes a function whose arguments are two real numbers (interpreted as coordinate values x and y), and its value is the represented dot.
- The set $C_{dot\ dot,\ line}$ = {*line*}, where symbol *line* denotes a function whose arguments are two dots and its value is the line segment defined between them.
- The set $C_{dot\ real,\ iline}$ = {*iline*}, where symbol *iline* denotes a function whose argumens are a dot and an angle -in radians- and its value is the infinite line passing through the dot and whose slope is equal to the tangent of the angle.

- The set $C_{line\ path,\ path}$ = {*path*}, where symbol *path* denotes a function whose arguments are a line and other path, and its value is the path resulting of concatenating the line and the path.
- The set $C_{path,\ polygon}$ = {*polygon*}, where symbol *polygon* denotes a function whose argument is a path and its value is a polygon if the path is closed.
- The set $C_{line,bool}$ = {*hor*, *ver*, *inc*}, where symbol *hor* denotes a function whose argument is a line and its boolean value indicates whether the line is horizontal. Similarly for symbols *ver* and *hor* for vertical and inclined lines, respectively.
- The set $C_{polygon,path}$ = {*path_of_polygon*}, where symbol *path_of_polygon* denotes a function whose value is the polygon interpreted as a path.
- The set $C_{real\ real\ real,\ dot3d}$ = {*dot3d*} constructs a 3D dot.
- The set $C_{dot3d\ dot3d,\ line3d}$ = {*line3d*} constructs a 3D line.
- The set $C_{line3d\ path3d,\ path3d}$ = {*path3d*} constructs a 3D path.
- The set $C_{path3d,\ polygon3d}$ = {*polygon3d*} construct a 3D polygon.
- The set $C_{view\ view\ dot,iline}$ = {*proj*}, where symbol *proj* denotes a function whose arguments are two views and a dot, and its value is the infinite line passing through the point that relates both views.
- The set $C_{iline\ line,\ dot}$ = {*intersection*} where symbol *intersection* denotes the following function: intersection(x_{iline},y_{line}*)* is the intersection dot of x_{iline} and y_{line}, if the dot belongs to A_{dot} , and $error_{dot}$ otherwise.
- The set $C_{iline\ iline,\ dot}$ = {*intersection*} defined similarly as *intersection* $\in$ $C_{iline\ line,dot}$.
- The set $C_{dot\ dot,\ real}$ = {*distance*} where symbol *distance* denotes a function whose arguments are two dots and its value is the distance between them.
- The set $C_{path,\ line}$ = {*head*} where symbol *head* denotes a function whose argument is a path and its value is the initial line in the definition of the path.
- The set $C_{path,\ path}$ = {*tail*} where symbol *tail* denotes a function whose argument is a path and its value is the path resulting from eliminating the first line of the argument path.
- The set $C_{dot\ polygon,\ bool}$ = {*dot_of_polygon*} where symbol *dot_of_polygon* denotes a function whose arguments are a dot and a polygon and its boolean value indicates whether the dot is on the boundary of the polygon.
- The set $C_{line\ polygon,\ bool}$ = {*line_of_polygon*} where symbol *line_of_polygon* denotes a function whose boolean value indicates whether the line belongs to the boundary of the polygon.

For practical purposes we define the control structure rule if-then-else as follows:

Additional syntactic rule:

7. If $\delta \in E_{bool}$, $\alpha \in E_s$ and $\beta \in E_s$ for some $s \in T$, then *if*(δ, α, β) $\in E_s$.

Additional semantic rule:

7. If $\delta \in E_{bool}$, $\alpha \in E_s$ and $\beta \in E_s$ for some $s \in T$, then

$$[[if(\delta, \alpha, \beta)]]^M = [[\alpha]]^M \quad \text{if } [[\delta]]^M = \mathbb{T}$$
$$[[if(\delta, \alpha, \beta)]]^M = [[\beta]]^M \quad \text{if } [[\delta]]^M = \mathbb{F}.$$

With this last definition we conclude the specification of our representational language. In the next section we show how expressions denoting graphical entities, relations and graphical concepts of descriptive geometry are formed and evaluated. In section 5 these expressions are used by the synthetic procedure which produces the solid model out of the orthogonal views.

4. Linguistic Representation of Graphical Concepts for the Interpretation of Orthogonal Views

In this section the graphical concepts that are required for the construction of the solid model of a polyhedron are presented.

A particular orthogonal drawing is represented by a set B_{dot} of expressions denoting dots, a set B_{line} of expressions denoting lines, a set $B_{polygon}$ of expressions denoting polygons and a set B_{iline} of expressions representing the vertical, horizontal and 45° axes.

Let *TOP* be the set of graphical entities in the top view.
Let *FRONT* be the set of graphical entities in the front view.
Let *RIGHT* be the set of graphical entities in the right-side view.
Let *VER* be the set of vertical lines.
Let *HOR* be the set of horizontal lines.
Let *INC* be the set of inclined lines.

4.1 Orthogonal Projections of a 3D Dot

As can be seen in Figure 4, a 3D dot always projects as a dot in any view. If the three orthogonal views are given, as in Figure 5, the 3D dot they describe is defined in Expression 2 which denotes a function from A_{dot} x A_{dot} x A_{dot} into A_{dot3d}. Graphical intities involved are shown in Figure 5.

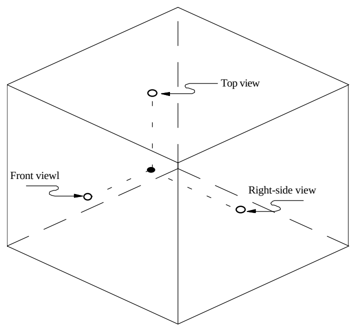


Figure 4 Orthogonal Projections of a Dot.

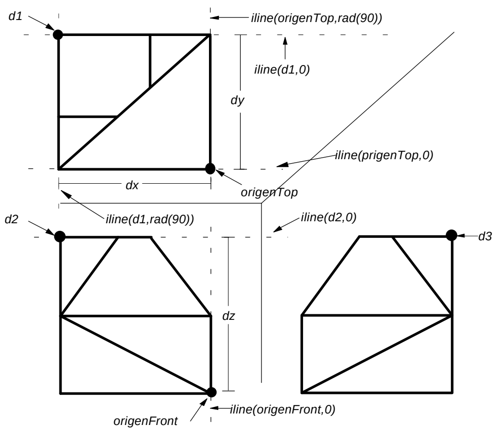


Figure 5 INTERPRETATION OF ORTHOGONAL PROJECTIONS OF A DOT. Distances *dx*, *dy*, *dz* are the coordinate values *x*,*y*,*z*, respectively, of the 3D dot. Dots o*rigenTop* and *origenFront* represent the origin of the tridimensional coordinate system.

Expression sort: *dot dot dot, dot3d*

$$
\begin{aligned}
&\boldsymbol{F_{dot3d}}\,(vertex) = \\
&[[\lambda x_{dot},\, y_{dot},\, z_{dot} \\
&\quad dot3d(distance(origenTop,\ intersection(iline(x_{dot},\ rad(90)),\ iline(origenTop,\ 0))), \\
&\qquad distance(origenTop,\ intersection(iline(x_{dot},\ 0),\ iline(origenTop,\ rad(90)))), \\
&\qquad distance(origenFront,\ intersection(iline(y_{dot},\ 0),\ iline(origenFront,\ rad(90))))) \\
&\ )]]^{M}
\end{aligned}
$$

Expression 2 3D DOT FROM THREE ORTHOGONAL PROJECTIONS. $x_{dot} \in TOP$, $y_{dot} \in FRONT$, $z_{dot} \in RIGHT$ are assumed. The three dots are assumed projections of a 3D dot.

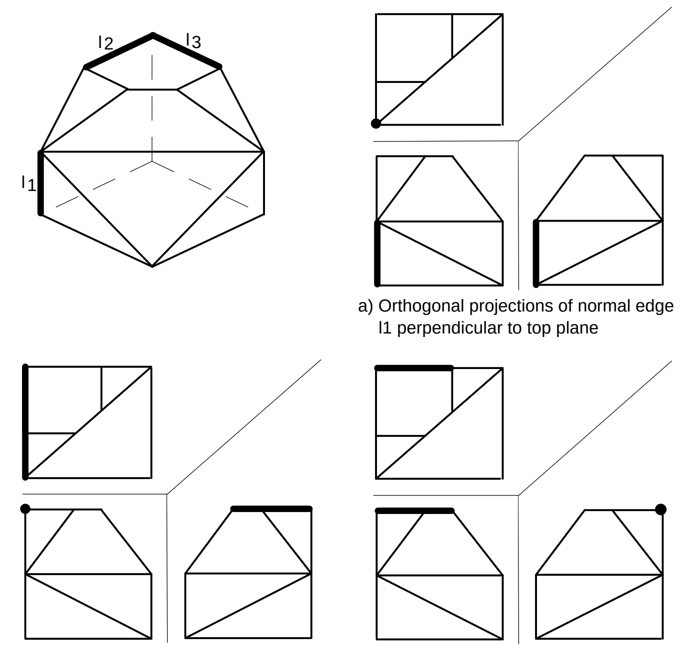
4.2 Orthogonal Projections of an Edge

An edge can be classified as *normal* edge, *inclined* edge or *oblique* edge depending on the relations it has with the planes of projection.

A *normal edge* is a line that is perpendicular to a plane of projection. It appears as a point in the plane of projection to which it is perpendicular and as a line in true length on adjacent planes of projection. In Figure 6 the three possible configurations of projection of a normal edge are illustrated.

An *inclined edge* is a line that is parallel to a plane of projection but inclined to adjacent planes. It appears as a line in true length on the plane to which it is parallel and foreshortened on adjacent planes. In Figure 7 the three possible configurations of the projections of an inclined edge are illustrated.

An *oblique edge* is a line that is oblique to all planes of projection. Since it is not perpendicular to any plane, it cannot appear as a dot in any view. It always appears as an inclined line in all views. In Figure 8 this case is shown.



a) Orthogonal projections of normal edge l1 perpendicular to top plane

b) Orthogonal projections of normal edge l2 perpendicular to front plane

c) Orthogonal projections of normal edge l3 perpendicular to right plane

Figure 6 Orthogonal projections of a normal edge.

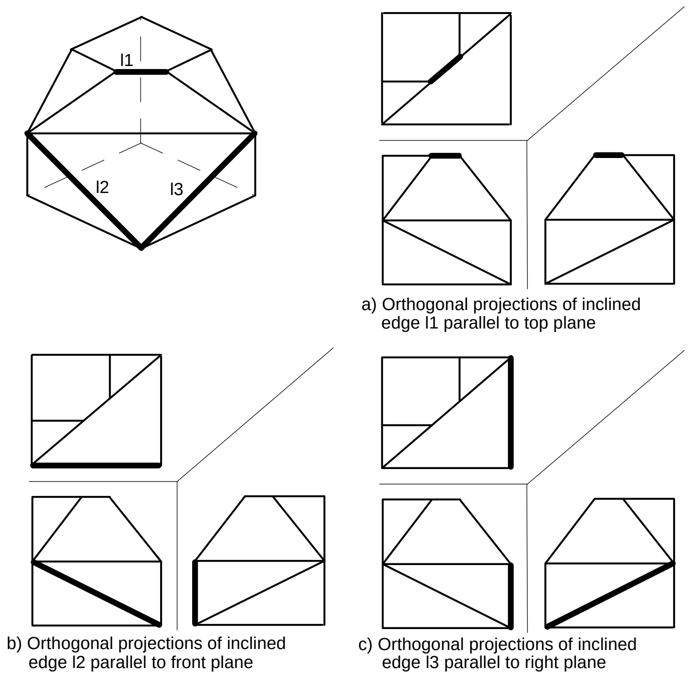

l1
l2
l3
a) Orthogonal projections of inclined edge l1 parallel to top plane
b) Orthogonal projections of inclined edge l2 parallel to front plane
c) Orthogonal projections of inclined edge l3 parallel to right plane

Figure 7 Orthogonal projections of an inclined edge.

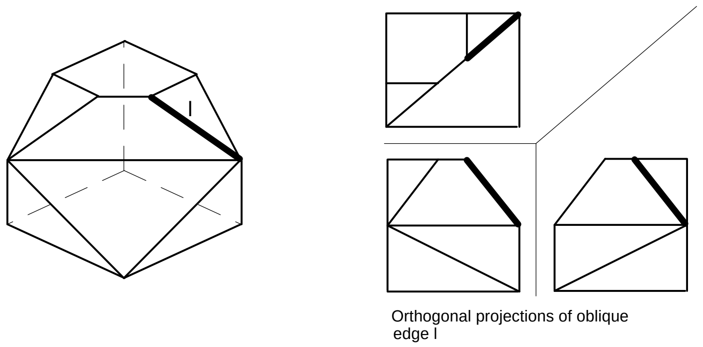


Figure 8 Orthogonal projections of an oblique edge.

As an example of the kind of expressions we model through the language L that involve the concepts above, consider Figure 9. The term defined in Expression 3 denotes a function from A_{line} x A_{line} x A_{line} into A_{line3d} that computes the 3D oblique line from its projections. In Figure 9 some subexpressions involved are shown.

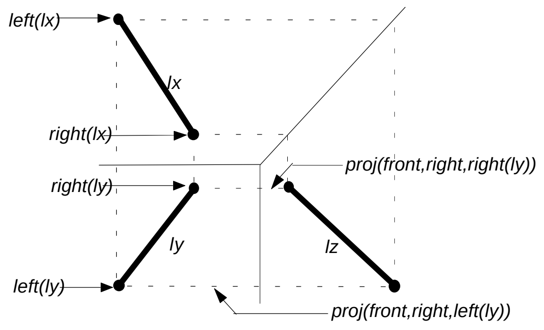


Figure 9 Interpretation of the orthogonal projections of an oblique edge.

Expression sort: *line line line, line3d*

$\boldsymbol{F}_{line3d}$ (*obliqueLine*) =

[[$\lambda x_{line}, y_{line}, z_{line}$ *line3d*(*vertex*(*left*(x_{line}), *left*(y_{line}), *intersection*(*proj*(*front*, *right*, *left*(y_{lline})), z_{line})),
vertex(*right*(x_{line}), *right*(y_{line}), *intersection*(*proj*(*front*, *right*, *right*(y_{line})), z_{line})))
]]^{M}

Expression 3 3D OBLIQUE LINE FROM ITS ORTHOGONAL PROJECTIONS. $x_{line} \in TOP \cap INC$, $y_{line} \in FRONT \cap INC$, $z_{line} \in RIGHT \cap INC$, are assumed. The three lines are assumed projections of a 3D line.

Similar expressions are defined for each of the three cases of normal and inclined edges.

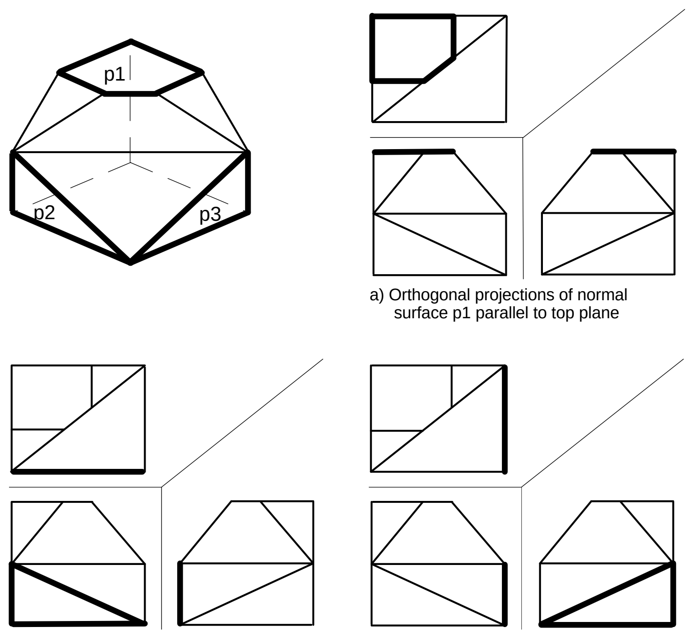
4.3 Orthogonal Projections of a Plane Polygonal Surface

A surface can be classified as *normal* surface, *inclined* surface or *oblique* surface depending on the relations it has with the planes of projection.

A *normal* polygonal surface is a plane polygonal surface that is parallel to a plane of projection. It appears as a polygon in true size and shape on the plane to which it is parallel, and as a vertical or horizontal line on adjacent planes. In Figure 10 the three possible configurations of the projections of a normal polygonal surface are illustrated.

An *inclined* polygonal surface is a plane polygonal surface that is perpendicular to one plane of projection but inclined to adjacent planes. An inclined surface appears as an inclined line on the plane to which it is perpendicular, and as a polygon on the others. In Figure 11 the three possible configurations of the projections of an inclined polygonal surface are illustrated.

An *oblique* polygonal surface is a plane surface that is oblique to all planes of projection. Since it is not perpendicular to any plane, it cannot appear as a line in any view. It always appears as a polygon in any view. In Figure 12 this case is illustrated.

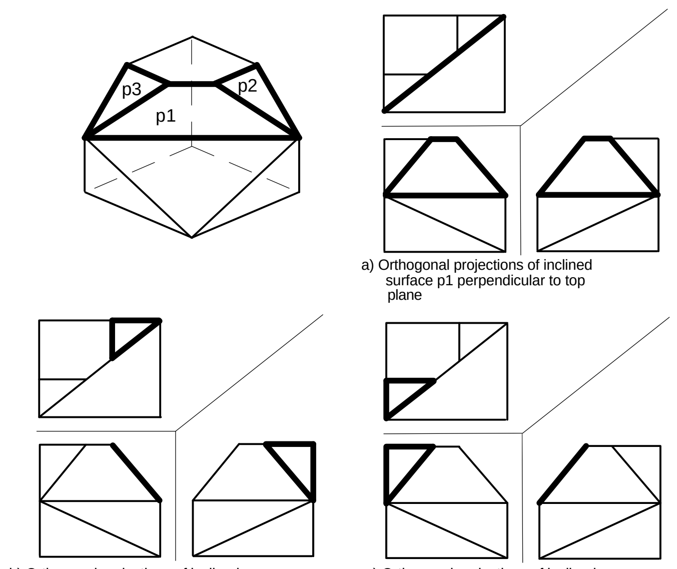


a) Orthogonal projections of normal surface p1 parallel to top plane

b) Orthogonal projections of normal surface p2 parallel to front plane

c)Orthogonal projections of normal surface p3 parallel to right plane

Figure 10 Orthogonal projections of a normal polygonal surface.



a) Orthogonal projections of inclined surface p1 perpendicular to top plane

b) Orthogonal projections of inclined surface p2 perpendicular to front plane

c) Orthogonal projections of inclined surface p3 perpendicular to right plane

Figure 11 Orthogonal projections of an inclined polygonal surface.

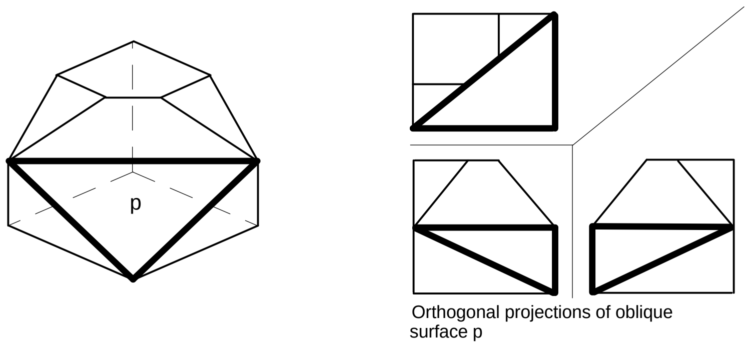


Figure 12 Orthogonal projections of an oblique polygonal surface.

For each of the 7 cases presented above (3 for normal surfaces, 3 for inclined surfaces and 1 for oblique surfaces) four expressions are defined. These four expressions are used in the process of constructing the solid model. In general, the first expression denotes a function whose arguments are two objects (lines or polygons) of different views and its boolean value indicates whether the two objects can be projections of a surface. The second expression denotes a function whose arguments are three objects (lines or polygons), one of each view, and its boolean value indicates whether they are projections of a surface. This expression assumes that two of the arguments actually can be projections of a surface (i.e. they have been already tested by the first expression). The third and fourth expressions denotes together a function whose arguments are three objects (lines or polygons), one of each view, and its value is the surface (3D polygon) whose projections are the arguments of the function.

As an example of the four expressions, consider the case of a normal surface parallel to the top plane illustrated in Figure 10a. In Figure 13 the conditions that objects in differents views must satisfy for being considered as projections of a normal surface within a drawing context are illustrated. The boolean function defined in Expression 4 has as its arguments a polygon in the top view and a line in the front view, and its value indicates if both objects can be projections of a normal surface parallel to the top plane, within a drawing context. What these terms evaluates can be summarized in the following phrase: *Each dot of the polygon has a corresponding -explicit- dot on the line*. In other words, each vertex of the surface must be projected as a dot on the line that is its front projection.

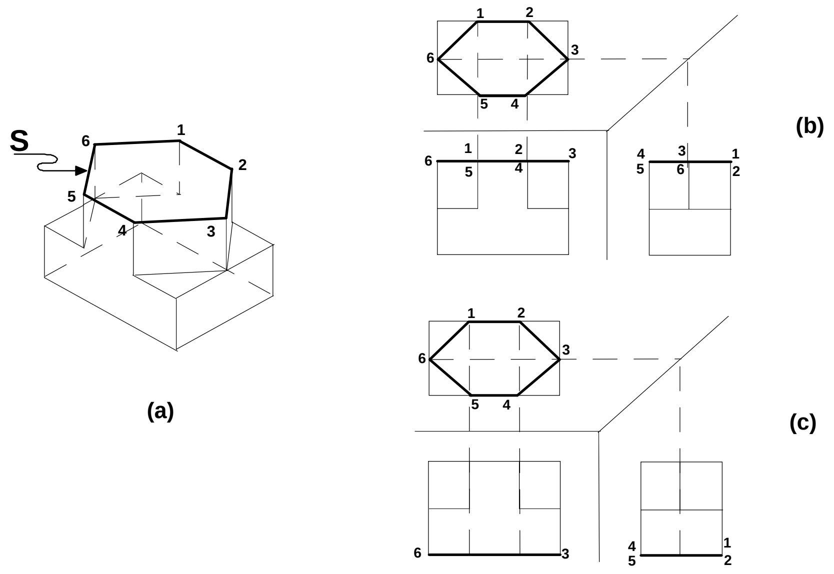


Figure 13 ORTHOGONAL PROJECTIONS OF A NORMAL SURFACE PARALLEL TO THE TOP VIEW WITHIN A CONTEXT. (a) Surface S. (b) The line defined from dot 6 to 3 in the front view can be a projection of surface *S* because all vertex of *S* are projected on it. (c) The line defined from dot 6 to 3 in the front view cannot be a projection of surface *S* because its vertex 1, 2, 5, 6 are not projected on it.

Expression sort: *polygon line, bool*

$$\boldsymbol{F_{bool}}\,(\mathit{polTop\_lineFront\ \_projectionsOf\_normalSurfParallelToTopPlane}) = $$
$$[\![\lambda x_{polygon}, x_{line}\ \ \forall w_{dot}\,(dot\_of\_polygon(w_{dot}, x_{polygon}) \rightarrow$$
$$intersection(proj(top, front, w_{dot}), y_{line}) \neq error_{dot}\ ]\!]^{M}$$

Expression 4 A POLYGON AND A LINE ARE PROJECTIONS OF A NORMAL SURFACE PARALLEL TO THE TOP PLANE. $x_{polygon} \in TOP$, $y_{line} \in FRONT \cap HOR$ are assumed.

Once a polygon in the top view and a line in the front view are evaluated for being projections, and given a second line in the right-side view, the boolean function defined in Expression 5 indicates whether the second line, together with the polygon and the first line, are projections of a normal surface. This term assumes that the first two arguments are projections and evaluates whether *the two lines are colinear and each dot of the polygon has a corresponding -explicit- dot on the line in the right-side view.*

Expression sort: *polygon line line,bool*

$$\boldsymbol{F_{bool}}\,(\mathit{polTop\_lineFront\ \_lineRight\_projectionsOf\_normalSurfParallelToTopPlane}) =$$
$$[\![\lambda x_{polygon}, y_{line}, z_{line}$$
$$colinear(y_{line}, z_{line}) \wedge$$
$$\forall w_{dot}\,(dot\_of\_polygon(w_{dot}, x_{polygon}) \rightarrow intersection(proj(top, right, w_{dot}), z_{line}) \neq error_{dot})\ ]\!]^{M}$$

Expression 5 A POLYGON AND TWO LINES ARE PROJECTIONS OF A NORMAL SURFACE PARALLEL TO THE TOP VIEW. $x_{polygon} \in TOP$, $y_{line} \in FRONT \cap HOR$, $z_{line} \in RIGHT \cap HOR$ are assumed. $x_{polygon}$ and y_{line} are assumed projections of a surface.

The term defined in Expression 6 denotes a function whose arguments are a polygon and two lines, and its value is the surface (3D polygon) whose projections are the arguments. This expression assumes that the three objects are projections of a surface. Expression 6 is basically formed by Expression 7.

Expression sort: *polygon line line, polygon3d*

$\boldsymbol{F_{polygon3d}}$ (*normalSurfaceParallelToTopPlane*) =
$[[\lambda x_{polygon}, y_{line}, z_{line}\ \ polygon3d(normalPathParallelToTopPlane(path\_of\_polygon(x_{polygon}), y_{line}, z_{line}))\ ]]^M$

Expression 6 NORMAL SURFACE PARALLEL TO THE TOP VIEW FROM ITS ORTHOGONAL PROJECTIONS. $x_{polygon} \in TOP$, $y_{line} \in FRONT \cap HOR$, $z_{line} \in RIGHT \cap HOR$ are assumed. $x_{polygon}$, y_{line} and z_{line} are assumed projections of a surface.

The term defined in Expression 7 denotes a function whose arguments are a path (actually, a polygon seen as a path) and two lines, and its value is the 3D path they represent. This term recursively evaluates the path line by line. To illustrate this evaluation process consider the simplest example of the projections of a normal surface shown in Figure 14. As can be seen, if the line in question -of the path- is vertical then the projections in front and right-side views must be a dot and a horizontal line, respectively, which correspond to the projections of a normal edge perpendicular to the front plane (Figure 6b). If the line is horizontal then the projections in front and right-side views must be a horizontal line and a dot, respectively, which correspond to the projections of a normal edge perpendicular to the right-side plane (Figure 6c). If the line is inclined then the corresponding projections in front and right-side views are two horizontal lines, which correspond to the projections of an inclined edge parallel to the top plane (Figure 7a).

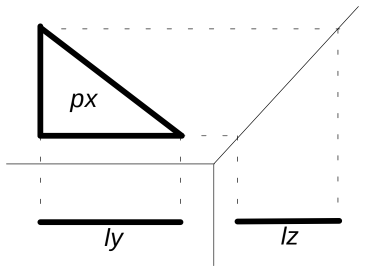


Figure 14 Projections of a normal suface.

Expression sort: *path line line, patht3d*

$\boldsymbol{F_{path3d}}$ (*normalPathParallelToTopPlane*) =
$[[\ \lambda x_{path}, y_{line}, z_{line}$
$\quad if\ (x_{path} = empty,$
$\qquad empty,$
$\qquad path3d(first3DLine(head(x_{path}), y_{line}, z_{line}), normalPathParallelToTopPlane\ (tail(x_{path}), y_{line}, z_{line})))\ ]]^M$

Expression 7 NORMAL SURFACE (AS A PATH) PARALLEL TO TOP PLANE FROM ITS ORTHOGONAL PROJECITONS. $x_{path} \in TOP$, $y_{line} \in FRONT \cap HOR$, $z_{line} \in RIGHT \cap HOR$ are assumed. x_{path}, y_{line} and z_{line} are assumed projections of a surface.

The symbol *first3Dline* denotes the function defined in Expression 8 (for clarity, we do not include the funcion defined in Expression 8 into Expression 7). This function has as its arguments the line in question of the evaluating path in the top view, and the two lines in the front and right-side views, and its value is the 3D line whose projections are the line in the top view -first argument- and two objects (a dot and a line or two lines) obtained from the lines in the second and third arguments. This function is based on the analysis above with Figure 14. Operator symbols *NormalEdgePerpendicularToFrontPlane*, *NormalEdgePerpendicularToRightPlane* and *InclinedEdgeParallelToTopPlane* denote functions of sorts A_{line} x A_{dot} x $A_{line} \rightarrow A_{line3d}$, A_{line} x A_{line} x $A_{dot} \rightarrow A_{line3d}$ and A_{line} x A_{line} x $A_{line} \rightarrow A_{line3d}$, respectively, whose values are edges of the sorts described in the names of the functions. These functions are defined in a similar way as the function in Expression 3.

```
                                                    Expression sort: line line line, line3d
F_line3d (first3DLine)=
 [[ λx_line, y_line, z_line
        if(ver(x_line),
          NormalEdgePerpendicularToFrontPlane(
                 x_line,
                 intersection(proj(top, front, up(x_line)), y_line),
                 line(intersection(proj(top, right, up(x_line)), z_line),
                      intersection(proj(top, right, down(x_line)), z_line))),
          if(hor(x_line),
            NormalEdgePerpendicularToRightPlane(
                   x_line,
                   line(intersection(proj(top, front, left(x_line)), y_line),
                        intersection(proj(top, front, right(x_line)), y_line)),
                   intersection(proj(sup, right, right(x_line)), z_line)),
            InclinedEdgeParallelToTopPlane(
                   x_line,
                   line(intersection(proj(top, front, left(x_line)), y_line),
                        intersection(proj(top, front, right(x_line)), y_line))),
                   line(intersection(proj(top, right, up(x_line)),z_line),
                        intersection(proj(top, right, down(x_line)), z_line))))) ]]^M
```

Expression 8 $x_{line} \in TOP$, $y_{line} \in FRONT \cap HOR$, $z_{line} \in RIGHT \cap HOR$ are assumed.

Similar expressions for six remaining cases of surfaces are defined in a similar way.

5. Generation of the Solid Model

In this section the process of building up the model of a polyhedron is illustrated with the help of an example. Consider the synthesis of the polyhedron in Figure 15b which is described by the views in Figure 15a.

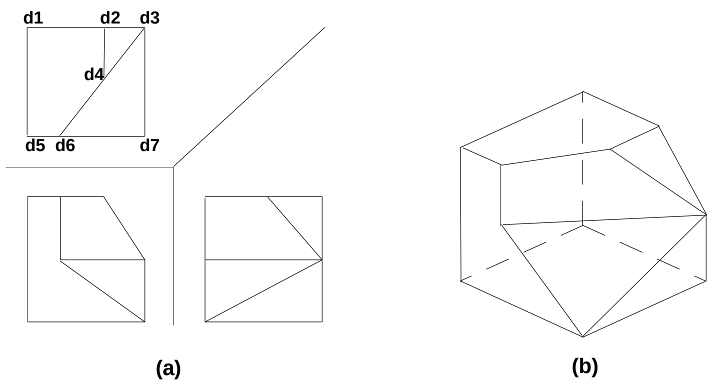


Figure 15 Views of a polyhedron.

For the construction process it is neccesary to distinguish between *visible* and *hidden* polygons. A polygon in a view is considered to be visible if it does not contain any other polygon inside it; for instance, the polygon defined by dots d1, d2, d4, d6, d5 in Figure 15a is visible. Otherwise, a polygon is considered hidden. The polygon defined by dots d1, d3, d7, d5 in Figure 15a (the bottom of the polyhedron) is an instance of the latter case. A hidden polygon is interpreted as a projection of an obstructed surface. A visible polygon can be interpreted as a projection of a non-obstructed surface, as a projection of an obstructed surface whenever projections of the obstructed and obstructing surfaces are the same polygon, or as a projection of surface that is partially obstructed. This last case is illustrated in Figure 16.

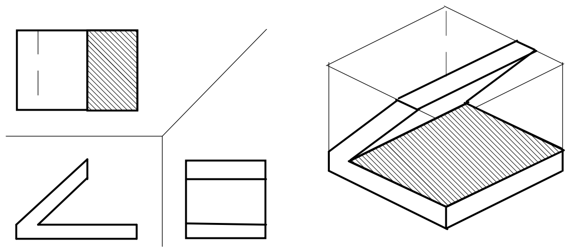


Figure 16 Partially obstructed surface.

The validity conditions of the kind of polyhedra that is considered in this paper were shown in Section 2 (illustrated in Figure 3), and are the following:

i) Each edge of a polyhedron must be shared exactly by two surfaces.
ii) Two adjacent surfaces of a polyhedron must not lie on the same plane.
iii) A surface of a polyhedron must be exactly one polygon.

For the construction of a polyhedron the following constraints are considered:

I) Each edge of a polyhedron must be shared exactly by two surfaces (condition *i*).
II) Two adjacent surfaces of a polyhedron must not lie on the same plane (condition *ii*).
III) Two surfaces of a polyhedron must not intersect.
IV) Two surfaces of a polyhedron must not overlap.
V) A suface whose projection on a view is a visible polygon is nearer to the plane of projection than any other surface whose projection on that view is a hidden polygon.
VI) A surface of a polyhedron must be exactly one polygon (condition *iii*).
VII) A surface is always either normal, inclined or oblique in relation to the planes of projection.

Condition *IV* can be considered as a particular situation of condition *II* because two overlapping surfaces can be seen as several adjacent surfaces lying on the same plane; This condition is explicitly included because the occurence of two overlapping surfaces is possible during the construction process. Condition *III* is included because two intersecting surfaces violate condition *I* or *VI*. Conditions *I* to *V* are verified during the construction process, while conditions *VI* and *VII* are implicitly considered in the procedure, as will be seen below.

In general, the construction process consists in obtaining a surface from the views and adding it to the set of surfaces of the 3-D model which is formed incrementally. Whenever a surface is added to the set, conditions *I* to *V* have to be satisfied. The process consists of four parts: First, for each visible polygon in the top view a surface is obtained according to condition *VII* (and considering condition *VI* implicitly), such that the set of all surfaces obtained in this way satisfies conditions *I* to *IV*. The second and third parts are similar for polygons in the front and right views, respectively. In the fourth part of the synthetic process the surfaces resulting from hidden polygons in the views or from visible polygons interpreted as obstructed surfaces are added, one at a time, verifying the validity conditions from *I* to *V*. When all edges of the constructed polyhedron satisfy condition *I* it must be verified that the orthogonal projections in the given three views correspond to the projections of the resulting polyhedron. This last verification assures that the method is sound.

As a notational convention we define the set of *maximal lines of a view* to be the largest set including all line segments which can be extracted from the view such that no two lines in the set satisfy the following two conditions: (1) they are adjacent and colinear and (2) they are colinear and overlapping. For instance, the set of maximal lines in the top view in Figure 15a is {line(d1,d3), line(d3,d7), line (d7,d5), line(d5,d1), line(d3,d6), line (d2,d4)}, but the lines line(d1,d2) and line(d2,d3) cannot be in the set for condition (1), and lines line(d1,d3) and line(d2,d3) for condition (2). Given an orthogonal drawing the sets *MLT*, *MLF* and *MLR* are the maximal lines sets for the top, front and right-side views, repectively.

Next, an overview of the construction procedure is presented. Consider *P* to be the set of surfaces of the polyhedron. *P* is initially empty.

Part I of the process consists in obtaining the surfaces resulting from visible polygons in the top view (i.e. projections of surfaces that are visible from the top plane). Once a polygon *p* is selected a surface *s* is sought by considering one of the following four cases (the fifth case considers that the polygon is a projection of a partially obstructed surface from that view –as shown in Figure 16):

T1. *s* is a normal surface parallel to the top plane (illustrated in Figure 10a).
T2. *s* is an inclined surface perpendicular to the front plane (illustrated in Figure 11b).
T3. *s* is an inclined surface perpendicular to the right plane (illustrated in Figure 11c).
T4. *s* is an oblique surface (illustrated in Figure 12).
T5. *s* is not a visible surface from the top plane (*s* is a partially obstructed surface).

These are all the cases where a surface is projected as a visible polygon in the top view and are graphically illustrated in Figure 17.

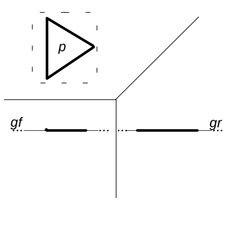
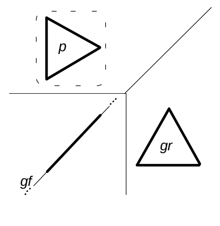
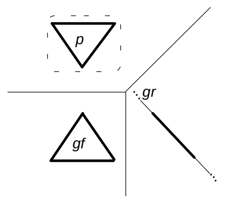
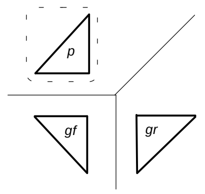
Normal Surface Parallel to Top Plane	**Inclined Surface Perpendicular to Front Plane**	**Inclined Surface Perpendicular to Right Plane**	**Oblique Surface**	**Not Visble Surface**
				The surface is considered partially obstructed where the polygon in the top view corresponds to the projection of a part of the surface. (see Figure 16).
RULE 1.a	**RULE 1.b**	**RULE 1.c**	**RULE 1.d**	**RULE 1.e**

Figure 17 Possible projections of a surface with a polygon in the top view. Labels at the bottom of drawings refer to the rules (defined in Figure 18) required to obtain the corresponding graphical objects in the front and right views.

For case T1 to hold two horizontal lines must appear as the front and right projections of the surface, as can be seen in the first case of Figure 17. These lines must be identified in the corresponding views. For this purpose a rule is defined for each of the four cases (the fifth case does not correspond to a visible surface). Rules 1a and 1b are shown in Figure 18 and the remaining rules are defined in a similar fashion.

RULE 1.a **Normal Surface Parallel to Top Plane**
Let p be a visible polygon in the top view.
Select g_f from $HOR \cap MLF$ (a horizontal maximal line in frontal view) such that
[[$polTop\_lineFront\_projectionsOf\_NormalSurfParallelToTopPlane(p, g_f)$]]=$\mathbb{T}$,
i.e. p and g_f can be projections of a normal surface.
Select g_r from $HOR \cap MLR$ (a horizontal maximal line in the right view) such that
[[$polTop\_lineFront\_lineRight\_projectionsOf\_NormalSurfParallelToTopPlane(p, g_f, g_r)$]]=$\mathbb{T}$,
i.e. p, g_f, g_r can be projections of a normal surface.

RULE 1.b **Inclined Surface Perpendicular to Front Plane**
Let p be a visible polygon in the top view.
Select g_f from $INC \cap MLF$ (an inclined maximal line in frontal view) such that
[[$polTop\_lineFront\_projectionsOf\_InclinedSurfPerpToFrontPlane(p, g_f)$]]=$\mathbb{T}$,
i.e. p and g_f can be projections of an inclined surface.
Select g_r from $C_{polygon} \cap RIGHT$ (a polygon in the right view) such that
[[$polTop\_lineFront\_polRight\_projectionsOf\_InclinedSurfPerpToFrontPlane(p, g_f, g_r)$]]=$\mathbb{T}$,
i.e. p, g_f, g_r can be projections of an inclined surface.

Figure 18 Examples or rules to obtain projections in the front and right views given a polygon in the top view.

As can be seen in RULE 1a, a horizontal maximal line g_f in the front view is chosen such that the polygon p in the top view and g_f are projections of a normal surface parallel to top plane. This is verified with the expression ($E1$) whose functional definition is shown in Expression 4:

($E1$) $polTop\_lineFront\_projectionsOf\_normalSurfParallelToTopPlane(p, g_f)$

If no line in the set $MLF \cap HOR$ satisfy ($E1$) case T1 as a whole does not hold, and another case is selected.

Once a line g_f in the front view is selected a horizontal maximal line g_r in the right-side view is selected in a similar way. Then it is verified that the three graphical objects can be projections of the surface evaluating the expression ($E2$) (for the definition of the functional operator see Expression 5):

($E2$) $polTop\_lineFront\_lineRight\_projectionsOf\_normalSurfParallelToTopPlane(p, g_f, g_r)$.

If expressions $E1$ and $E2$ hold for the graphical objects p, g_f and g_r then the surface s is obtained. This is done evaluating the expression corresponding to the case of a normal surface parallel to top plane: (see definition in Expression 6):

($E3$) $normalSurfaceParallelToTopPlane(p,g_f,g_r)$.

Similar expressions are employed for the definition of rules 1b to 1d.

For every surface s obtained by this procedure if the set $P \cup \{s\}$ satisfy constraints from I to IV s is included in P. In case a surface s cannot be included in P through case T1, cases T2 to T4 are tried out. If none of these cases hold the current polygon is a projection of a partially obstructed surface (case T5 is assumed and no surface is obtained from polygon p).

At this stage of the computation, all surfaces that can be extracted from visible polygons in the top view are included in *P*. Parts II and III of the process obtain surfaces visible from the front and right views which were not obtained before in a similar fashion.

The cases for visible polygons in the front view are (Part II):

F1. *s* is a normal surface parallel to the front plane (illustrated in Figure 10b).
F2. *s* is an inclined surface perpendicular to the top plane (illustrated in Figure 11a).
F3. *s* is an inclined surface perpendicular to the right plane (illustrated in Figure 11c).
F4. *s* is an oblique surface (illustrated in Figure 12).
F5. *s* is not a visible surface from the front plane (*s* is a partially visible surface).

And the cases for visible polygons in the right view are (Part III):

R1. *s* is a normal surface parallel to the right plane (illustrated in Figure 10c).
R2. *s* is an inclined surface perpendicular to the top plane (illustrated in Figure 11a).
R3. *s* is an inclined surface perpendicular to the front plane (illustrated in Figure 11b).
R4. *s* is an oblique surface (illustrated in Figure 12).
R5. *s* is not a visible surface from the right plane (*s* is a partially visible surface).

Part IV of the synthetic procedure consists in identifying the surfaces that are hidden from the three orthogonal views. This procedure consists in finding out the surfaces required to close the current polyhedron. For this purpose all edges of the polyhedron (formed with the surfaces in set *P*) that belong exactly to one surface are identified as the set *R*. As each edge must belong exactly to two surfaces according to condition *I*, it is required to find an additional surface for each edge in *R*. For each edge its three orthogonal projections are computed and the result of this computation is matched against the views. For this process it is important to highlight that all hidden polygons of all the three views that have the reference edge as one of its sides must be considered. The projections of the edge must be included in the projections of the surface sought (i.e. the edge must bound the surface).

For instance, consider the case of a normal edge perpendicular to top plane (l_1 in Figure 6a). This kind of edge can only bound surfaces of the three following kinds:

- normal surface parallel to front plane (Figure 10b)
- normal surface parallel to right plane (Figure 10c)
- inclined surface perpendicular to top plane (Figure 11a)

These cases are illustrated in the first row of Figure 19 where the projections of the edge are enclosed with dashed lines. If the projections of an edge are of kind L1 in Figure 19 then one of the cases S1 to S3 must hold. The case that matches identifies the projections of a surface and as a consequence the surface itself. If the resulting surface satisfies conditions *I* to *V* (condition *V* is satisfied if no visible surface –obtained in Parts I, II and III of the process– is obstructed by the surface obtained) it can be included in *P*. We repeat this procedure until all edges in *R* are shared by two surfaces. If all edges in *R* are tried out but not all of them are shared by exactly two surfaces another solution must be sought. If there is no solution which satisfy the conditions the three given views do not correspond to a valid polyhedron.

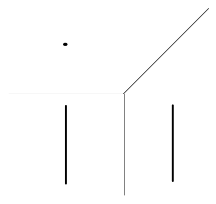
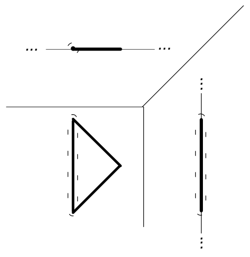
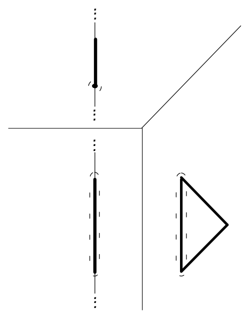
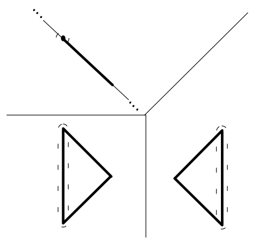
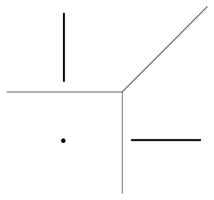
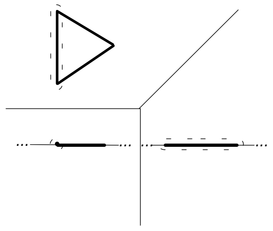
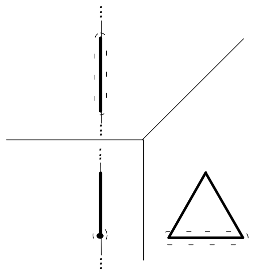
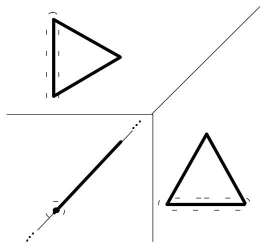
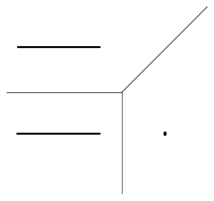
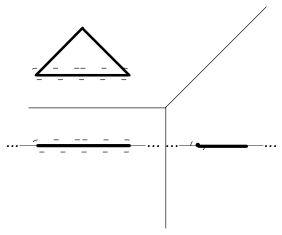
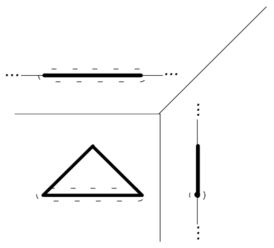
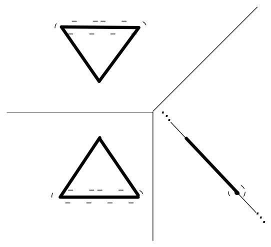
	KIND OF SURFACE		
KIND OF EDGE	**S1**	**S2**	**S3**
L1 **Normal Edge Perp. to Top Plane** 	**Normal Surface Parallel to Front Plane** 	**Normal Surface Parallel to Right Plane** 	**Inclined Surface Perp. to Top Plane** 
L2 **Normal Edge Perp. to Front Plane** 	**Normal Surface Parallel to Top Plane** 	**Normal Surface Parallel to Right Plane** 	**Inclined Surface Perp. to Front Plane** 
L3 **Normal Edge Perp. to Right Plane** 	**Normal Surface Parallel to Top Plane** 	**Normal Surface Parallel to Front Plane** 	**Inclined Surface Perp. to Right Plane** 

Figure 19 Cases where a *normal* edge can bound a surface.

The synthetic process of the 3D model is exemplified with the generation of the polyhedron in Figure 15b. The construction procedure for the visible polygons (Parts I to III) is shown in the tree search space in Figure 20. When this process is completed, the process mentioned above to include the hidden surfaces is carried out. As can be seen, the process is quite simple and the procedure can handle the construction of solid models of the kind defined above in Section 1 and 2. Further details of the specific construction rules are given in [Garza95].

Finally, it can be mentioned that as the construction rules take into account all possible cases in which a surface can be generated, and that conditions *I* to *VII* rule out all configurations of surfaces that do not conform to a valid polyhedron, we can be confident that the method is sound. For the same reason if a set of projections determine a valid polyhedron this will be included in the search space and subsequently we can be confident that the procedure is complete. The search space, however, might be very large. A gross measure of it in terms of the number of surfaces can be approached if it is considered that the maximum depth of the space is the same as the number of surfaces, and the branching factor at the main decision level (i.e. whether a surfaces is included or not) is four because given a polygon in one of the views there are only four possible kinds of surfaces of which the polygon is a projection. However, given the constraints imposed by conditions *I* to *VII* this exponential figure can be significantly reduced.

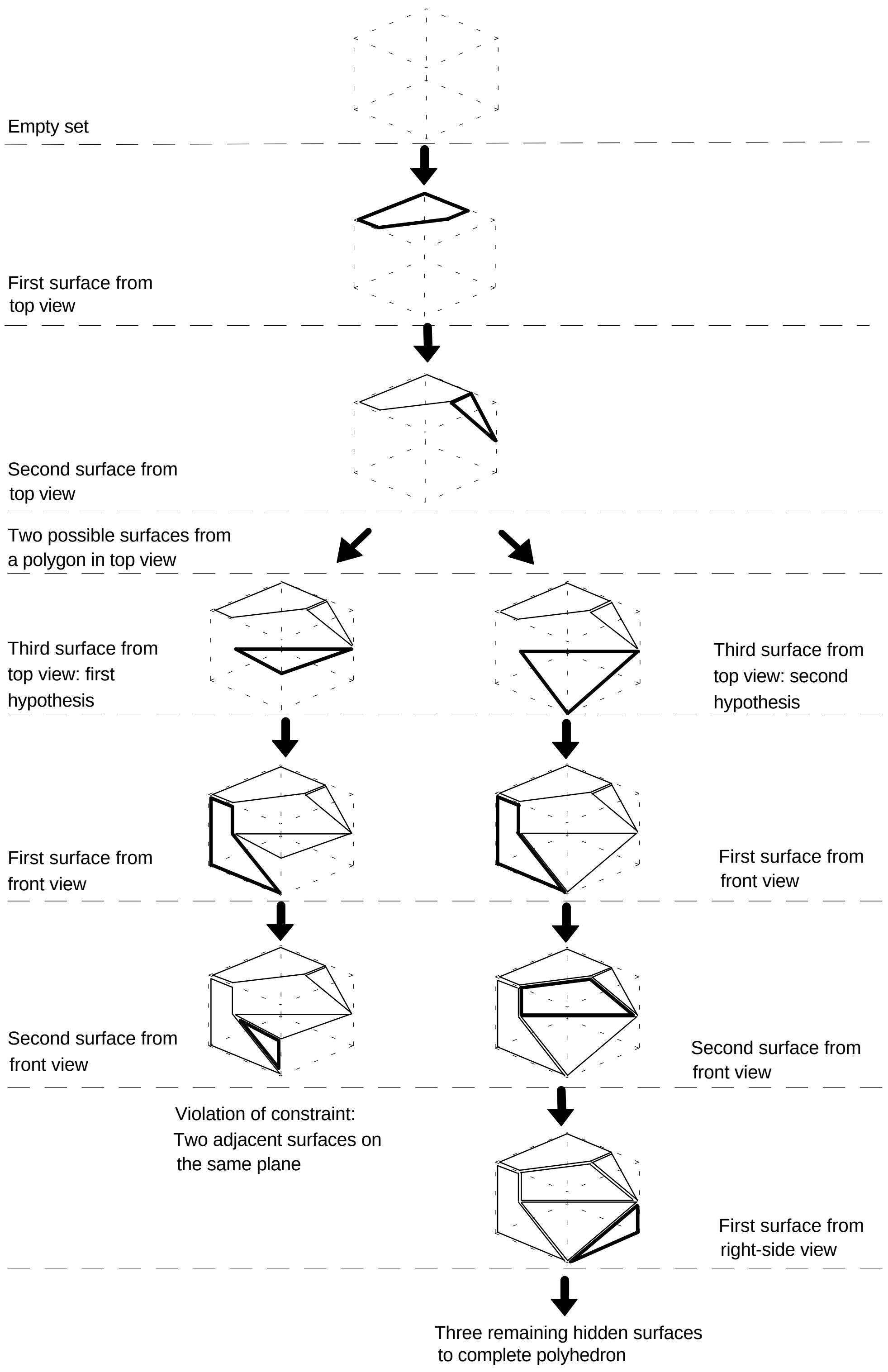

Empty set
First surface from top view
Second surface from top view
Two possible surfaces from a polygon in top view
Third surface from top view: first hypothesis
Third surface from top view: second hypothesis
First surface from front view
First surface from front view
Second surface from front view
Second surface from front view
Violation of constraint: Two adjacent surfaces on the same plane
First surface from right-side view
Three remaining hidden surfaces to complete polyhedron

Figure 20 Construction process of the polyhedron shown in Figure 15.

6. Conclusions

In this paper a procedure for constructing 3D models of polyhedra from their orthogonal views has been presented. While previous approaches to the problem are based on quantitative algorithms, the qualitative method presented here resembles intuitive reasoning processes used in human drafting practices. The construction procedure was designed to model a graphical reasoning process with natural constraints. To support the task a representational language expressive enough for modeling drafting concepts and procedures of descriptive geometry has been presented. The language is a multi-sorted first-order logical language with equality augmented with functional abstraction and extends the graphical an logical representational languages developed with the GRAFLOG system [Pineda89,91,92]. It has been shown how graphical entities and relations constituting the orthogonal views can be represented through expressions of the language. Similarly, the expressions representing graphical concepts required for the interpretations of the views in the construction of the solid model have been presented. Finally a construction procedure for the production of the solid which resembles intuitive processes employed in human drafting practices has been illustrated. The logical language and its interpretation process has been fully implemented. All drafting concepts for the interpretation of 3D dots, lines and polygons out of the 2D entities in the orthogonal views have also been tested. The final construction procedure outlined in Section 5 is currently being implemented. The present research is a case of study for the more general problem of reasoning with graphical representations. Our results support the hypothesis that graphical concepts can be represented in an abstract fashion through the lambda calculus allowing us to contemplate the integration of graphical and linguistic concepts in a common representational formalism.

References

[Dowty81] D. R. Dowty, R. E. Wall, S. Peters. "Introduction to Montague Semantics". D. Reidel Publishing Company, Dordrecht, Holland, 1981.

[Garza95] E. G. Garza “Síntesis de poliedros a partir de sus vistas ortogonales: un caso de estudio en razonamiento gráfico”, Msc. Thesis, ITESM Campus Morelos, 1995.

[Goguen78] J. A. Goguen, J. W. Thatcher, E. G. Wagner. "An Initial Algebra Approach th the Specification, Correctness and Implementation of Abstract Data Types". *Current Trends in Programming Methodology*, ed R.t. Yeh, vol IV, pp. 80-149, Prentice-Hall. 1978.

[Haralick82] R. M. Haralick, D. Queeney. "Understanding Engineering Drawings". *Computer Graphics and Image Processing*, 20:3, pp. 242-525. 1982.

[Idesawa73] Masanori Idesawa. "A System to Generate a Solid Figure from Three View". *Bull. JSME* 16, pp. 216-225. 1973.

[Lafue76] Giles Lafue. "Recognition of Tthree-dimensional Objets from Ortographic Views". *Computer Graphics*, Vol. 10, No. 2. 1976.

[McCarthy77] McCarthy.

[Nagendra88] I. V. Nagendra, U. G. Gujar. "3-D objects from 2-D orthographic views -a survey". *Computer and Graphics* Vol 12, pp. 111-114. 1988.

[Pineda89] L. A. Pineda. "GRAFLOG: A Theory of Semantics for Graphics with applications to Human-Computer Interaction and CAD Systems", 1989. Ph.D. thesis, University of Edinburgh.

[Pineda92] L. A. Pineda. "Reference, Synthesis and Constraint Satisfaction", *Eurographics'92 Conference Proceedings*, Cambridge, U. K. 1992.

[Preiss84] K. Preiss. "Constructing the solid representation from engineering projections". *Computers and Graphics* 8, pp 381-389. 1984.

[Sakurai83] H. Sakurai, D. C. Gossard. "Solid model input through orthographic views". *ACM/SIGGRAPH* 17, pp. 243-252. 1983.