Comput. & Graphics Vol. 11, No. 4, pp. 427–443, 1987
Printed in Great Britain.

0097-8493/87 $3.00 + .00

Technical Section

A LINGUISTIC APPROACH TO 3-D OBJECT RECOGNITION

WŁODZIMIERZ KASPRZAK
Institute of Computer Science, Polish Academy of Sciences, 00-901 Warsaw, P.O. Box 22, Poland

Abstract—The principles of a 3-D object recognition system for combined intensity-image and depth-map understanding are discussed. The goal of such system is to be an inversion of image synthesis performed by 3-D computer graphics. A linguistic model for two system elements, the knowledge base and recognition strategy, being an extension of pattern recognition approaches, is outlined. It consists of a powerful object specification language and a simultaneous syntactic-semantic analysis in this language. The syntax is based on a node-controlled parallel structure grammar. Particular attention is paid to elements shared in common by several parts and to hidden line/surface problems. Both are embedded into the grammars derivation. The semantics is well-defined due to the attribution of the grammar.

INTRODUCTION

Three-dimensional object recognition is a fundamental task performed by computer vision systems[1]. Many researchers have not yet presented justification of their methods in logical and/or mathematical terms. Such a theory is known for two-dimensional pattern recognition. Two classes of pattern recognition methods can be distinguished[2–3]: decision-theoretical and structural. The former one arises from a few branches:

- Syntactic approach[4]
- Linguistic approach[5]
- Logical approach[6]

Most image understanding research performed in the past has concentrated on using digitized intensity images as sensor data. Such object recognition is an ill-posed problem because light-source parameters and surface reflectance functions are not known. It is expensive or impossible to overcome the lack of information arising from the 3-D scene to 2-D image projection.

In recent years, though, digitized range data have become available due to the following:

- Stereo vision[7–8]
- Range finders (especially laser devices)[9–10].

Range data are often produced in the form of an array of numbers referred to as a range-image (or depth map), where the numbers quantify the distance from the sensor focal plane to object surfaces within the field of view.

Object recognition using range image is still difficult, but is a well-posed problem when observable object shapes are known. Subsystems for this task can be distinguished, which can be independently modelled. A close relation to computer graphics tasks, better than before, can be established.

1. THE OBJECT-RECOGNITION PROBLEM

The real world that we see and touch is primarily composed of solid materials. Those materials which we handle on the high-level as primitives of knowledge representation I define as OBJECTS. In general as OBJECTS, I consider assemblies of SOLIDS with defined structure.

For any given object we can define the origin of its local coordinate system. Each object occupies space, and at most one object can occupy any given point in space. For reference purposes we assume the existence of a world coordinate system related to the range finder location, which can be placed at any convenient position. Objects are positioned in space relative to this coordinate system by means of translation, rotation and axis scaling. I denote the 9 transformation parameters as "pos."

Moreover, with each object we associate a vector of qualitative features. They are of two sorts:

- Structure attributes (denoted "ats")—describing the objects composition from solids
- Feature attributes (denoted "atf")—describing the texture, colour or other surface characteristic

The triple ("pos," "ats," and "atf") can be called object ATTRIBUTES.

Let "Ncl" be the number of objects distinguishable by the system. We refer to the i-th object as "ob_i." The number of instances of that object I denote as "N_i." The OBJECT SPACE is the set of ordered pairs:

$$W = \{(ob_i, (\text{pos}, \text{ats}, \text{atf})_{ij})\, I \quad i = 0, 1, \ldots, \text{Ncl}; j = 1, \ldots, N\}$$

where $(\text{pos}, \text{ats}, \text{atf})_{ij}$ is the vector of attribute values for the j-th instance of object ob_i. The additional object "ob_0" is the sensor object with position "pos_0" and feature-vector "atf_0."

The set of all positions is contained in the set $\text{Real}^3 \times \text{Real}^3 \times \text{Real}^3$, since the domains of translation, rotation and scaling belong to Real^3. The domains for all structure and feature attributes are finite or at most enumerated.

A depth sensor obtains a depth-map projection of a scene. We model this projection as a mathematical operator, D, which maps elements in the set W_Domain = Objects × Positions × Structure_Domains × Feature_Domains into elements in

the set of all scalar functions of 2 variables, which we denote as Depth_f:

$$D{:}W\_\text{Domain} \rightarrow \text{Depth}\_f$$

$$\text{Depth}\_f(x) = g/ob, \text{pos}, \text{ats}/(x) = D(ob, \text{pos}, \text{ats}),$$

where x is the vector of 2 variables of the focal plane of the sensor. This notation demonstrates that the set of depth-map functions associated with a single object is an infinite family of functions, when the set of positions is not enumerated, and an enumerated family of functions, if opposite.

Since objects do not occupy all input elements, we need a convention for depth-map function values for this spatial vector x values, which do not correspond to object surface points. If the point $(x, \text{Depth}\_f(a))$ can not lie on an object surface, we assign the value of $-\infty$ to Depth_$f(x)$. Hence we write the projection of a set of m objects as follows:

$$\text{Depth}\_f(x) = \max_{1 \le i \le m} g/(ob, \text{pos}, \text{ats})_i/(x)$$

$$= \max_{1 \le i \le m} D((ob, \text{pos}, \text{ats})_i).$$

Note that the single-object occlusion operator is included in the depth operator, D, and that the multiple-object occlusion is performed by the max computation.

For understanding intensity images additional object space elements are needed. Besides the sensor object that receives light, I introduce an object that generates light—"*ol*," with position—"pol" and feature attributes—"atl." A surface reflectance characteristic—"ref"—must be further appreciated for each pair "object, feature attribute values." The intensity image is given by applying an illumination-reflectance operator "INT":

$$\text{Int}\_f(x)$$

$$= \text{INT}\{[\max_{1 \le i \le m} D((ob, \text{pos}, \text{ats})_i), \text{atf}_i, \text{ref}_i], ol, \text{pol}, \text{atl}\}$$

to the depth-map function, by given light source and feature attributes and reflectance characteristics of objects.

The intensity-image object recognition problem is generally much more difficult than depth-map recognition, owing to the additional inversion of the INT operator. But when solved, it gives also the feature attributes of the recognized object instance. This is a helpful information for high-level processing.

Hence the combined depth-map and intensity-image object recognition problem can be formulated as:

"Given an intensity image, Int_$f(x)$, associated with the depth-map function, Depth_$f(x)$, by operator INT, determine the sets of possible objects with corresponding positions, structure and feature attributes, that could be projected under given intensity-reflectance of objects and light source characteristics to obtain Int_$f(x)$ and Depth_$f(x)$."

2. OBREC SYSTEM

2.1 *Principles*

A proposal to handle data of three abstraction levels in a computer vision system was recently offered in [11–12]—LOW-level, OBJECT-level and HIGH-level.

Low-level processing includes image formation, processing and segmentation. It fulfills a transformation from real scene into a symbolic description that uses only local-surface primitives.

Object-level processing uses symbolic description coming from the low-level as input data. It performs surface reconstruction, volume localization and object description.

High-level processing gives interpretations in terms of natural language or the same level of abstraction.

But until now most researchers mean that high-level knowledge is needed to build interpretations on lower levels. In fact, in traditional intensity-image understanding one can hardly speak about a homogeneous object-level. The knowledge about light and physics allows us only to hypothesize 3-D cues from the image. But interpretation is achieved due to the high-level world model.

The 2-D and 3-D descriptions can not be obtained from the image data unambiguously by bottom-up processing without the help of the high-level. I claim that range data allows to overcome this problem and to split the world-model into one for the object-level and the other for high-level. On both levels image interpretation can be performed, although on object-level it is a single-image interpretation and on high-level a dynamic interpretation.

Now the high-level expectations about what is seen control the effectiveness of lower processes, but do not decide in primitive extraction and interpretation problems. This control permits the following examples:

- More than one object-level interpretation can exist; the high-level chooses among them.
- Refinement of object-interpretation—more details in image can be analyzed.
- Extension of object-interpretation—other image places, not only those centrally located, will be interpreted.

To eliminate possible problem ambiguities, I define precisely the OBJECT RECOGNITION SYSTEM (OBREC system) as an object processing level of a computer vision system with at least intensity-image and depth-map as input data. It means that the signal input consists at least of an array of pixel intensities and a corresponding array of pixel depths. Any other corresponding array of numerical values can be available.

2.2 *Input data*

The input data to the OBREC system is created by low-level segmentation processes and consists of local-surface-description primitives:

- Homogeneous surface patches—achieved by region growing methods
- Contour lines—as a result of edge detection methods
- Vertices—given by applying local image operators

2.3 *Knowledge base*

The main idea of system performance is that one operates on symbolic descriptions represented as AT-

TRIBUTED RELATION STRUCTURES. I distinguish three parts of the knowledge base:

- The syntax and semantics of ENTITIES and RELATIONSHIPS between them—from this elements the SCENE DESCRIPTIONS are built.
- Knowledge about dependences between entities and relationships given as so called BACKGROUND RULES.
- DERIVATION SCHEME—a finite method of deriving descriptions one from another.

2.3.1 *Scene descriptions.* By an ENTITY I mean a class of scene description primitives. An ENTITY is defined by two schemas:

- $E1$—vector of ATTRIBUTES:
  - Geometry functions which specify the location of entity in 3-D space
  - Feature functions giving texture, colour and geometric characteristics
- $E2$—set of CONSTRAINTS:
  - Independent restrictions for attribute values (for example "little cylinder" means that the attribute "radius" is not longer than 1 m).

A RELATIONSHIP between entities is defined by one scheme:

- R—N-ARY EXPRESSION constraining the attributes of entities (for example, a relationship can represent an expression which is invariant under linear space transformations).

How are the entities and relationships represented? I represent them as LABELS of structure-nodes or structure-hyperarcs (the last I call n-EDGES if the relationship is n-ary). The MEANING of labels is in accordance with the schemas $E1$, $E2$ and R. But we need a language to express these schemas. This goal is accomplished by the first-order predicate language—LS. Hence I distinguish the syntax and semantics of definition schemas for entities and relationships besides the entity- and relationship-representations and meanings.

An ENTITY INSTANCE occurs if a semantics for LS is given, which values the vector of entity attributes, and these values satisfy the entity constraint set.

A RELATIONSHIP INSTANCE occurs if a semantics for LS is given, which values attributes of all entities being in this relationship, and these values satisfy the n-ary expression defining the relationship. Particular entity and relationship-INSTANCES are represented by the following:

- ⟨node identifier⟩:⟨node label⟩—entity instance
- ⟨n-edge identifier⟩:⟨n-edge label⟩ (⟨node 1 identifier⟩:⟨node 1 label⟩, . . . , ⟨node n identifier⟩:⟨node n label⟩)—relationship instance.

The node- and n-edge-identifiers belong to the set of Normal numbers $\mathcal{N}$. Hence we handle an enumerated set of entity- and relationship-instances. Now the correspondence between the meta-syntax (in terms of structure elements) and the syntax of schemas must be established in order to define the meaning of an entity instance undirectly by the semantics of its LS-representation.

The attributes of an entity instance represented as "k:A" are $t^1/x_k^1, \ldots, t^n/x_k^n$, where the $t^i - s$ are terms representing expressions which value the variables $x_k^i - s$ indexed by identifier "k."

The CONSTRAINTS are represented as LS-formulas over the attribute-variables, $x_k^i, \ldots, x_k^n$.

An n-ary EXPRESSION for relationship instance is represented as an LS-formula over the attribute variables of all entity instances specified by ⟨node 1 identifier⟩, . . . , ⟨node n identifier⟩.

An entity TOPOLOGY HIERARCHY must be distinguished (as shown in Table 1, for example). I underline the introduced hierarchy so that each entity can be fully defined by a set of entities and relationships of lower levels (Fig. 1 gives an example).

Each finite set of entity instances from the same topology level and relationship instances among them is called SCENE DESCRIPTION and is represented by an attributed structure.

2.3.2 *Rules.* A BACKGROUND RULE relates two attributed structures. I want two representation schemas for 3-D solids in CAD-systems to be generalized by these rules:

- The boundary representation (for volumes)
- The constructive solid geometry (for constructing complex solids as assemblies of volumes)

One rule, which relates the meta-syntax, is called PRODUCTION rule and the other one, which relates the LS-elements associated with structures, is called SUBSTITUTION rule. A production is of the following form:

$$B \leftarrow K, \quad K' \rightarrow B',$$

where B is the left and B' the right structure isomorphic class; K, K' are substructure classes, respectively. The substitution rule has components of type t/x, where t is a term in LS, and x is a variable representing an entity-attribute. For all entity instances in B and B', such components are given.

Some of the entities due to their substitution components are called DEFINING. Only variables for defining entities can occur in the terms of this substitution rule. The substitution for defining an entity is each x/x, where x stands for entity-attributes.

Table 1. Topology levels of OBREC system

Level	Kind of entities used
6	objects
5	solids
4	volumes (shells)
3	faces
2	loops (closed edges) and homogeneous areas
1	edges and elementar flat hom. areas
0	vertices, line segments and elementar flat h. areas

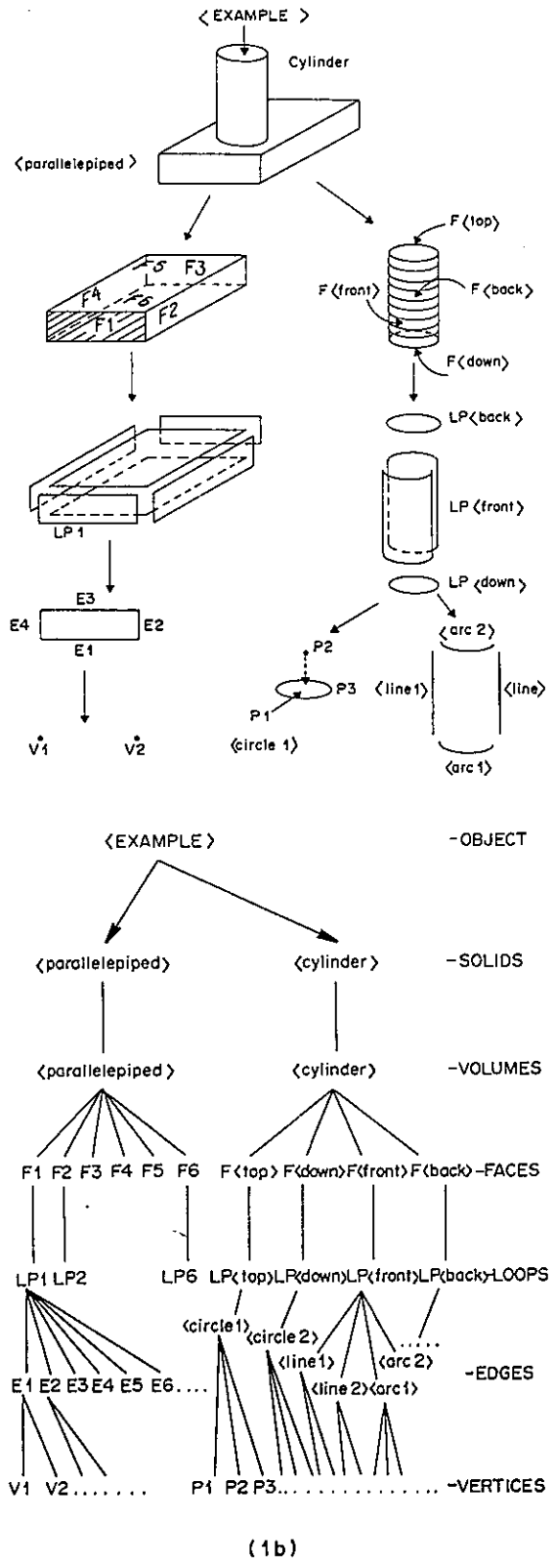


Fig. 1. Object description-levels: (a) object "EXAMPLE," (b) hierarchy of entities describing "EXAMPLE."

If the entities from the left description are being interpreted on the base of the right description-entities, they are called SYNTHESIZED entities (interpretation in a bottom-up manner). If some entities of the right description are being interpreted, they are called DERIVED entities (top-down interpretation).

2.3.3 *Derivations.* The DIRECT DERIVATION is a transition between two descriptions. To obtain a derivation $G \Rightarrow G'$ one must:

1. Establish a family of structures $B1, B2, \ldots, Bn$, such that G is the union of the $Bi - s$ $(i = 1, \ldots, n)$. Intersections $K_{ij} = Bi \cap Bj$ $(1 \leqslant i \leqslant j \leqslant n)$ will arise.

2. Select productions $P = \{B_i \leftarrow k_{ij}, k'_{ij} \rightarrow B'_i\}$ $(1 \leqslant i \neq j \leqslant n)$ with $k_{ij} = k_{ji}$, $k'_{ij} = k'_{ji}$ for $j < i$.

3. Each substructure K_{ij} of G is replaced by K'_{ij} $(1 \leqslant i < j \leqslant n)$ and each substructure B_i of G is replaced by B'_i $(i = 1, \ldots, n)$, such that $K'_{ij} = Bi' \cap Bj'$ and K_{ij}, K'_{ij} are isomorphic with k_{ij}, k'_{ij}.

4. The structure G' is the union of the $Bi' - s$ $(i = 1, \ldots, n)$.

5. Using the substitution rules associated with productions P, if only the defining entities are interpreted, all other entities of Bi-, $Bi' - s$ can be interpreted.

A formal definition of the syntax part of derivation is given in [13]. The derivation scheme is illustrated by Fig. 2 (for $n = 3$). Two proper subsets of labels must be distinguished—STARTED and TERMINAL. CONFIGURATION is each derivation beginning with a description having only started labels and ending with a description having only terminal labels.

2.3.4 *Restrictions.* I restrict my interest to those class of configurations where each configuration can be decomposed in several STEPS.

A transition between descriptions from neighbour levels is called the CONSTRUCT step. Rules used in this derivation can have substitutions for synthesized entities only, or both synthesized and derived. The second case occurs, if on the reached level a so called "visibility" step will be performed.

Rules used in a CONSTRUCTi step have only substitution for synthesized entities. This step is internal to one level and is an iterative constructing of descriptions by the same subset of rules. Rules, that have only substitutions for derived entities, are used in steps internal to one level:

- Similar entities gluing (GLUING step)
- Approximation of "complex" entities (APPROXIMATION step)
- Hidden entities elimination (VISIBILITY step)

I call the steps given above REFINEMENT steps, because they relate semantically equal descriptions, deriving more details of them. Each configuration is de-

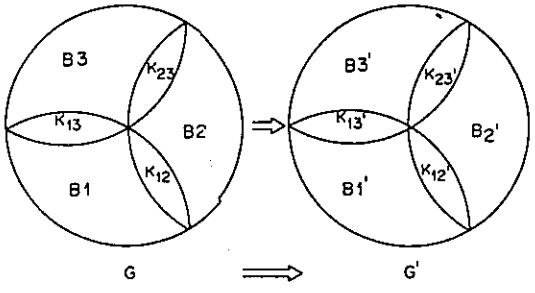


Fig. 2. Derivation scheme.

Table 2. Types of topology-levels

Level	Type	Refinement steps performed
5	ALFA0	CONSTRUCT GLUING VISIBILITY (elimination of complete hidden solids)
4	BETA	CONSTRUCT GLUING iterative CONSTRUCT
3	ALFA0	CONSTRUCT GLUING VISIBILITY (elimination of faces hidden by other faces belonging to the same volume)
2	ALFA	CONSTRUCT GLUING APPROXIMATION (of curved homogeneous areas) VISIBILITY (elimination of hidden flat hom. areas)
1	ALFA	CONSTRUCT GLUING APPROXIMATION (of curved edges) VISIBILITY (elimination of hidden line edges)
0	ALFA0	CONSTRUCT GLUING

composed in steps performed on levels of three types. A level is called type BETA, if an iterative CONSTRUCTi step is allowed, but no visibility- and approximation-steps occur. Levels without iterative CONSTRUCTi step are called type ALFA or simpler case ALFA0 (there is no approximation step) (see Table 2).

Some auxiliary rules with only synthesized entities must be given, which allow partly to inverse the refinement steps. I call them GENERALIZATION rules. They are not used by the derivation scheme, but represent necessary conditions to interpret the higher level entities on the base of lower level descriptions.

This generalization rules perform the following:

- Establishing of relationships among a set of entity instances (CREATE)
- Inversion of approximation (APPROX_INV)
- Inversion of entity gluing (GLUING_INV)

The inversion of entity hidding is performed by substitutions for synthesized entities used in CONSTRUCT step.

2.4 *Recognition strategy*

The recognition's GOAL is to interpret a given input set in terms of object instances. The METHOD used is a linguistic analysis—one must achieve a configuration over the input (sub-)set.

A two-directional syntactic-semantic analysis is performed (Fig. 3):

1. Interpretation of all synthesized entities of the configuration, by using generalization rules and CONSTRUCT-parts of substitutions for synthesized entities (bottom-up analysis).

2. Top-down derivation with derived entities-interpretation and synthesized entities-verification.

The verification means that some of the descriptions used in the first analysis must be included in appropriate descriptions from the second one. These are descriptions related by CONSTRUCT (CONSTRUCTi) steps and CREATE rules. The analysis steps on a particular level are pictured in Fig. 4.

Now we explain a strategy for achieving the recognition's goal. Imperatives of a recognition strategy in an OBREC system are the following:

- Multiple ambiguous interpretations of input data can be achieved.
- There need not be an interpretation of all input data.

Partly and ambiguous interpretations are allowed, since OBREC is only a subsystem and the complete recognition is controlled by the high-level processing.

I define the OBREC-recognition strategy as consisting of two sequential processes ($T1$, $T2$) and one cyclical process ($T3$) (Fig. 5).

$T1$ is a hypothesis generation process. All interpretations of possible synthesized configuration-entities on the base of generalization rules and synthesized parts of CONSTRUCT steps are achieved. The applied synthesized parts in CONSTRUCT(i) steps and entity instances related by it built a so-called HYPOTHESIS NET with object instances being the highest layer.

$T1$: Input_Description →

HYPOTHESIS_NET with $H(6) = (ob_1, \ldots, ob_K)$.

$T2$ is an aggregation of object instances into a family of (partly) object interpretations.

$$T2: \quad H(ob_1, \ldots, ob_K) \rightarrow S(I_1, \ldots, I_L), I \subset 2^H.$$

$T3$ is a search in the space of achieved interpretations for finding a best one. One chooses a best interpretation first and constrains the entity instances building a Start_description for top-down refinement ($T31$). Then

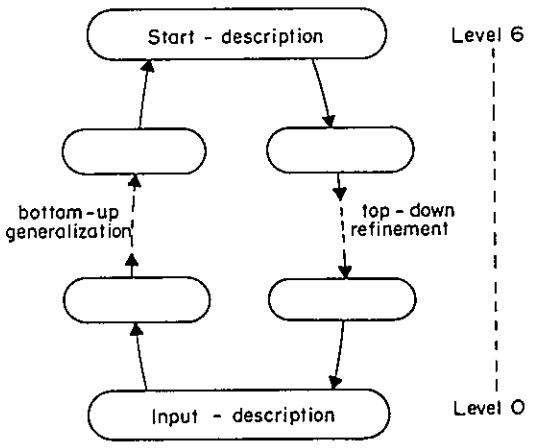


Fig. 3. Two-directional analysis.

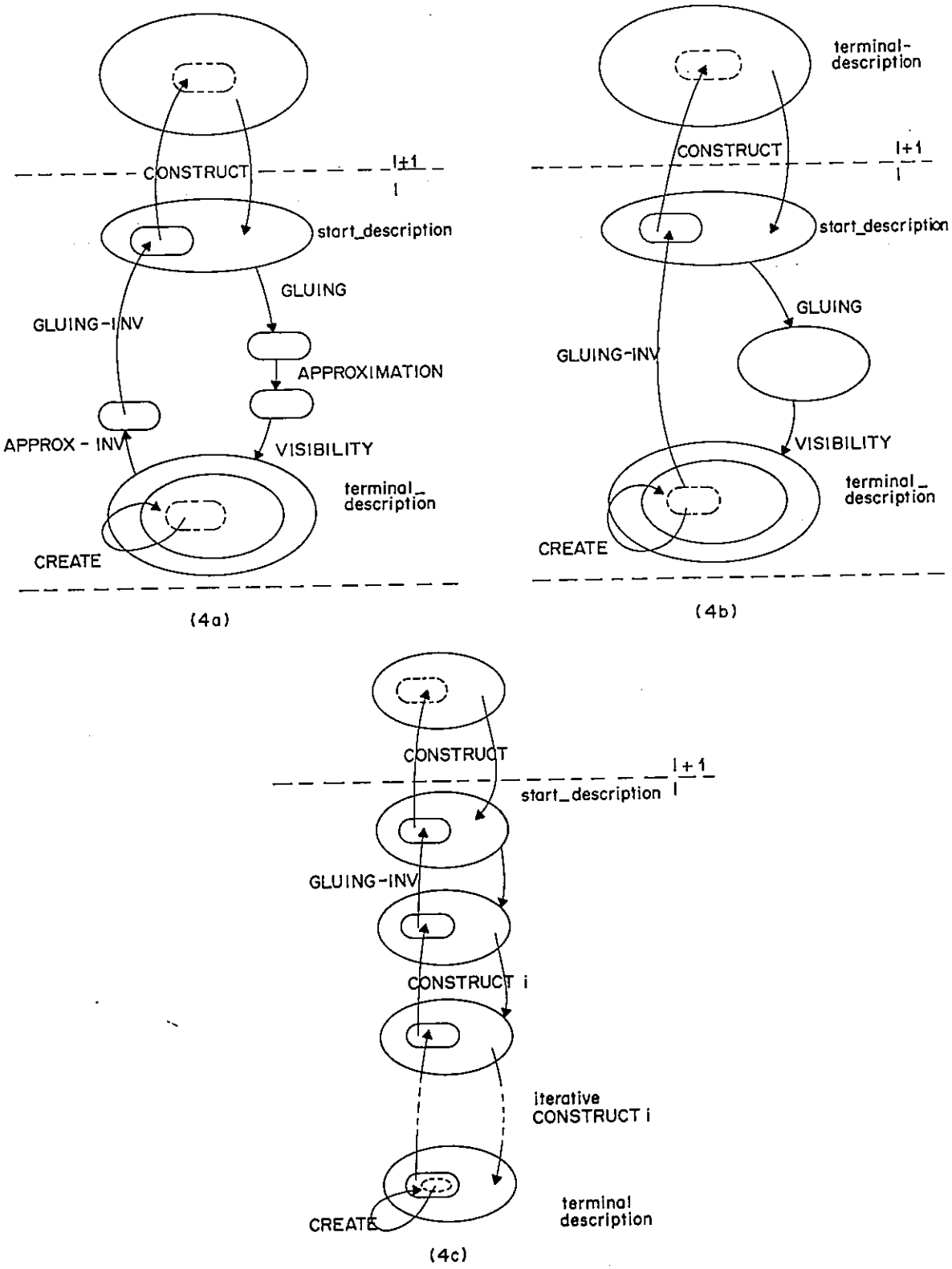


Fig. 4. Analysis on level I of type: (a) ALFA0, (b) ALFA, (c) BETA.

a derivation process is performed, controlled by appropriate hypothesis sub-net ($T32$). If a proper configuration is established, the best interpretation is found. If not, the interpretation space will be modified and the cycle started again ($T33$).

$$T3: \quad S(I_1, \ldots, I_L) \rightarrow I.$$

All elements of the OBREC-knowledge base will be described in an Object Specification Language (OSL). The recognition strategy will act as a linguistic analysis of OSL-expressions.

3. THE OSL-GRAMMAR

The ordered pair (g_A, g_E), with g_A being a finite set of node-identifiers and g_E being a finite set of identifiers for n-edges between nodes $(2, 3, \ldots, N)$ from g_A, is called RELATION STRUCTURE of level N (short: N-structure).

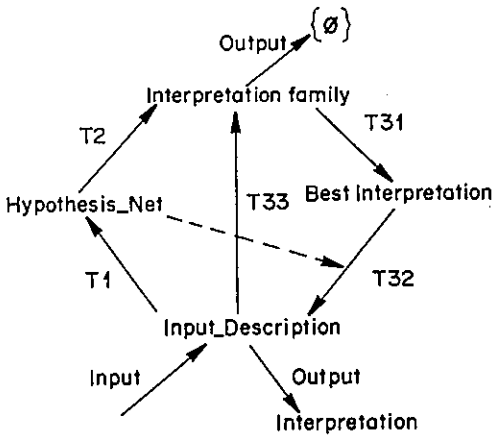


Fig. 5. Recognition strategy.

Let (V_A, V_E) be a pair of finite label sets, g be some N-structure, (m_A, m_E) be a pair of mappings—m_A: $g_A \to V_A$, m_E: $g_E \to V_E$. The pair $(g, (m_A, m_E))$ is a labeled relation structure, which I call *VN*-structure. Let VNSTR denotes the set of all *VN*-structures. The base of OSL syntax is a node-controlled parallel structure grammar.

3.1 *Node-controlled parallel structure grammar* (*nPSG*)

Now I give an extension of graph grammars[14–15]. A production will be defined as a triple:

(left node, right substructure; insert rule).

It relates nodes of input structure with substructures of output structure, and determines how n-edges between left nodes of productions applied in the direct derivation can be replaced by n-edges between right substructures.

A set of BASE OPERATORS must be given first:

$$T = \{L_A | A \in V_E\} \cup \{R_A | A \in V_E\}.$$

By OPERATORS I mean words over the alphabet:

$$OP = V_A \cup T \cup \{(,), C, I, \bar{}\},$$

such that T is the set of base operators; if X is an operator, then CX, aX, $\bar{X}$ for $a \in U_A$ also are; if X, Y are operators, then XY, $X \cup Y$, $X \cap Y$ also are.

Definition 1. Let $H \in$ VNSTR. If O is an operator and $a \in H_A$, then the NEIGHBORHOOD $O(a)$ means the following:

1. $O = L_A$ or $O = R_A$—the neighborhood is a set of node-chains arriving in n-edges with label A, from which the n-edge comes to node a or to which the n-edge goes from node a.
2. $O = C\Lambda$—chains of nodes from H_A which do not belong to $\Lambda(a)$.
3. $O = v_1, \ldots, v_m \Lambda(v_j \in V_A^*, i = 1, \ldots, m)$—node-chains from $\Lambda(a)$, with node-labels specified at least by one of $v_1, \ldots, v_m$.
4. $O = \Lambda \cap \Gamma$—node-chains which belong to $\Lambda(a)$ and $\Gamma(a)$.
5. $O = \Lambda \cup \Gamma$—node-chains which belong to $\Lambda(a)$ or $\Gamma(a)$.
6. $O = I$—all nodes from H.
7. $$O = \bar{\Lambda} = \begin{cases} I, & \text{if } \Lambda(a) = \varnothing \\ O, & \text{if } \Lambda(a) \neq \varnothing \end{cases}.$$
8. $O = \Lambda\Gamma$—node-chains from neighborhood $\Lambda(\underline{a})$ of each $\underline{a} \in \Gamma(a)$.

Without point (7) the definition of neighborhood is standard.

Definition 2. NODED STRUCTURAL production is a triple, $(B_l, B_r; S)$, where $B_l, B_r \in$ VNSTR and B_l has only one node; $S = \{S_{e_1}, \ldots, S_{e_k}\}$ is an INSERT rule such that

$$S_{e_i} = \{l_{e_i}, r_{e_i}\} \quad (i = 1, \ldots, k), \quad e_i \in V_E,$$

$$l_{e_i} = \bigcup_{j=1}^{p_i} (O_j(a); V_{A_j}), \quad r_{e_i} = \bigcup_{j=1}^{a_i} (V_{B_j}; O_j(a))$$

are called input- and output-inserting COMPONENTS; $O_j(a)$ is the neighborhood of node $a \in B_l$; V_{A_j}, V_{B_j}—chains of nodes from B_r.

Clearly a production is APPLICABLE to some node "i:A" ($i \in \mathcal{N}$) only if the production's left node is labeled by A.

Example 1. Let a *VN*-structure G and a production, such as pictured in Fig. 6 are given. The neighborhoods defined in structure G are then:

$$dR_A(1) = \{2\}, \quad L_B(1) = \varnothing, \quad R_B(1) = \{23\}.$$

Definition 3. H' is DIRECTLY DERIVABLE from H ($H \to H'$) by the set of noded structural productions, $P(V)$.

There exist productions, $p_1, \ldots, p_k$, isomorphic with productions from $P(V)$ and

- $H_A = \bigcup_{m=1}^{k} a_m$—set of left nodes;

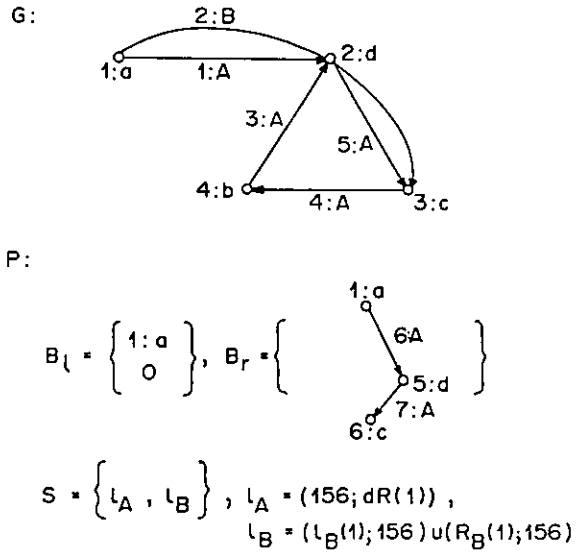


Fig. 6. Example of a noded structural production.

- $H'_A = \bigcup_{m=1}^{k} A_m$—set of right-substructure nodes;
- $H'_E = \left\{\bigcup_{m=1}^{k} B_m\right\} \cup B'$,

where B' is a set of inserted n-edges, defined as follows:

$$e'(a_1, \ldots, a_p) \in B' \quad (m'_E(e') = E \in V_E, a_i \in H'_A (i = 1, \ldots, p)) \Leftrightarrow$$

There is a decomposition of $(a_1 \cdots a_p)$—$X_1 \ldots X_S$—and "s" insert rules with the S_E-components: $S_E^1, \ldots, S_E^S$ having

$$r_E = (X_j; A_{E_j}(a^j)) \in S_E^j, l_E = (B_{E_j}(a^j); X_j) \in S_E^j,$$

such that

for $s = 1$:

$$a_j^1 \in A_E(a^1), \quad a_j^1 \in B_E(a^1);$$

for $s > 1$:

$$\text{exist } Y_{R_j} \in A_{E_j}(a^j), \quad Y_{L_j} \in B_{E_j}(a^j) \quad (Y_{R_j} \neq a^j, (Y_{L_j} \neq a^j) \quad (j = 1, \ldots, s)$$

and

$$-Y_{L_1} \cdot \{a^1\} \cdot Y_{R_1} = \cdots = -Y_{L_S} \cdot \{a^S\} \cdot Y_{R_S} \quad (Y_{L_i} = Y_{R_S} = \emptyset).$$

- H' has no multiple n-edges.

Direct derivation is a relation between two equational classes of VN-structures. Figure 7 shows a simple derivation scheme.

Definition 4. A NODE-CONTROLLED parallel structure grammar (nPSG) is a triple $(SP, R^S, \rightarrow)$, where $SP = (V^N, V^T, P, V^S)$ is a system of noded structural productions with non-terminal (N), terminal (T) and start (S) labels, $R^S = \bigcup_i R^i$ is a finite family of started VN-structures with the same node set and $m_j(R^i_A) = V^S_A$, and $\rightarrow$ is a direct derivation by productions from P.

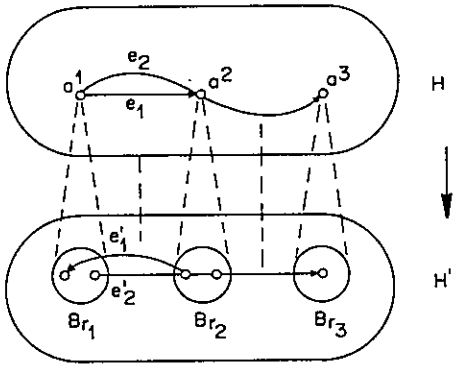


Fig. 7. A simple direct derivation scheme.

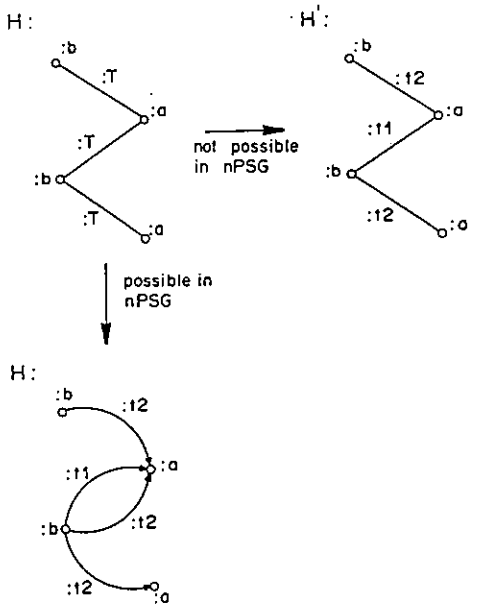


Fig. 8. Example of direct derivation-limitations in nPSG.

How usual the set of SENTENCES of some grammar $\mathcal{T} \in$ nPSG is: $L(\mathcal{T}) = \{H \in V^T\text{NSTR} | R^i \in R^S, R^i \rightarrow^* H\}$, where "$\rightarrow^*$" is the reflexive and transitive closure of relation "$\rightarrow$."

nPSG over the alphabet V is called COMPLETE if for all node labels from V at least one applicable production exist. The completeness assures that each derivation is carried on.

nPSG is called MONOTONICAL if all productions have proper right substructures.

nPSG is called SYNCHRONIZED if terminal labels appear only in the last direct derivation, when sentences are derived. It can be proved that for each nPSG an equal synchronized nPSG exist. Equality of grammars means both generate the same set of sentences.

Definition 5. A pair (p, K), where p is a noded structural production and $K \in$ VNSTR is called production with APPLICATION CONTEXT. (p, K) is APPLICABLE in $H \in$ VNSTR to node "a" iff p is applicable to "a" and there is no $k \cong K$, such that $a \in k$ and $k \subset H$.

The application context specifies some node-neighborhood conditions for applying a production. It is in no terms similar to the context for string grammars.

It can be proved, that for each nPSG over monotonical productions with application context an equal nPSG exists. I have proved also[16] that for each nPSG over productions with application contexts an equal nPSG exists, if nonmonotonical productions are applied only to nodes, which are neighbors of at least one node being replaced by a proper substructure. This condition will be satisfied in our applications.

3.2 *Virtual nPSG—CONSTRUCT step*

nPSG, being a base of attributed grammar, is not always effective. One is often not able to distinguish between different instances of the same entity (different nodes having the same label) for replacing n-edges. For example as shown in Fig. 8, we can not derive from

structure H the structure H'. The smallest derivable structure in nPSG, which includes H', is H''. We have to define a wider class of grammars, in some sense non-deterministical replacing n-edges.

Definition 5. Let $G \in$ VNSTR, $a \in G_A$, O—neighborhood operator. If the neighborhood $O(a)$ is nonempty, then the VIRTUAL neighborhood of node a in G (denoted $O\omega(a)$) is the set of $O(a)$-subsets.

A production with at least one virtual neighborhood is called VIRTUAL production.

Definition 6. A VIRTUAL direct derivation is each direct derivation, if exactly one element from each virtual neighborhood was chosen in order to insert n-edges.

The CONSTRUCT step will be performed by virtual productions. Depending on the semantics of nodes, a semantic rule will determine exactly one value for each used virtual neighborhood.

3.3 *The GLUING- and VISIBILITY-steps*

Let us distinguish a subset of edge labels called terminal IDENTITY labels and denote them as follows: $ID = \{IDA | A \in V_A\}$.

Definition 7. Let $B \in$ VNSTR; $a_1, a_2 \in B_A$; $m_A(a_1) = m_A(a_2) = A$; $o(a_1, a_2) \in B_E$; $m_E(o) = IDA$. The GLUING of nodes a_1, a_2 IN B (denoted $\mathrm{GLU}_A[B(a_1, a_2)]$) is a structure, D, defined as follows:

$$D_A = B_A - \{a_1, a_2\} \cup a; m_A(a) = A;$$

D_E consists of n-edges from B_E, where the nodes a_1, a_2 were replaced by node a, multiple n-edges were reduced to one of them and all n-edges passing both a_1, a_2 were simultaneously removed.

Each gluing operation represented by a terminal identity edge can be "simulated" equal up to isomorphism by a derivation step in nPSG. But to achieve that from structures derived by applying a set of noded productions, only these are "proper," which simulate the gluing operations, we have to use productions with some application conditions.

Two gluings $\mathrm{GLU}_A[B(a_1, a_2)]$, $\mathrm{GLU}_A[B(a_3, a_4)]$ are called INDEPENDENT, if at most their first nodes are the same. Gluings represented by edges with various labels are always independent. Two and more independent gluings can be simulated parallel in one derivation step.

Definition 8. Let $D \in$ VNSTR. The STEP of GLUING of nodes in D is each composition of all gluings represented in D (denoted $\mathrm{GLU}(D)$).

It is easy to see that the composition of gluings is transitive, hence the step of gluing nodes defines some class of isomorphic VN-structures. One can prove what follows:

Proposition 9. There exists a set of noded structural productions with application contexts, which "simulates" equal up to isomorphism the gluing step, if the input structure satisfies the following restriction:

"In each clique, due to edges labeled from ID, there is exactly one node, which is not a second node of any gluing operation; and there are no multiple edges."

Structures we will deal with satisfy the restriction given above. Now let us distinguish an another set of edge labels called HIDING labels (denoted as HID), having terminal (HID^T) and nonterminal (HID^N) parts. A function

$$f\_\mathrm{hid}: \quad V_A \times V_A \to 2^{\mathrm{HID}^T}$$

gives "proper" labeling dependent from the node-labels of the edge being labeled.

Definition 10. Let $H \in$ VNSTR; $i, j \in H_A$; $m_A(i) = A$, $m_A(j) = B$; $o_t(j, i) \in H_E$, $m_E(o_t) = T \in f\_\mathrm{hid}(A, B) \in \mathrm{HID}^T$.

The VISIBILITY of node i specified by edge o_t in H is an operation giving $D \in$ VNSTR, defined as follows (denoted $D = \mathrm{VIS}_T[H(i{:}a - j{:}b)]$):

$$D_A = [H_A - \{i\}] \cup \{i_1, \ldots, i_L\}; \quad m_A(i_j) = A_j$$

$$(j = 1, \ldots, L) \quad \text{and } L \text{ depends upon } (T, A, B).$$

$$D_E = [H_E - \{o_t\} - H'_E] \cup H''_E \cup H'''_E,$$

where

$$H'_E \subset H_E - \{o_t\} = n\text{-edges having node "}i\text{"};$$

$$H''_E \subseteq \bigcup_{k=1}^{L} \{e'_k | e \in H'_E, m_E(e) \notin \mathrm{HID}^T$$

and node "i" is replaced by "i_k"};

$$H'''_E \subseteq \bigcup_{k=1}^{L} \{e'_k | e \in H'_E, m_E(e)$$

$\in \mathrm{HID}^T$ and node "i" is replaced by "i_k"};

$$m_E(e'_k) = \begin{cases} m_E(e), & \text{if } e'_k \in H'^j_E \\ T', & \text{if } e'_k \in H'''_E \quad (T' \in f\_\mathrm{hid}(A, A_k) \text{ for } e'_k(i_k, i). \end{cases}$$

The operation defined above is nondeterministic, since it specifies a family of possible structures D. Each n-edge in H having the first node of VIS operation, can be replaced by at most L n-edges, maybe various labeled.

Example 2. Let $\langle hta \rangle \in V_A$ specifies a homogeneous triangle area.

$f\_\mathrm{hid}(\langle hta \rangle, \langle hta \rangle) = \{t1, \ldots, t9\}$ is the set of possible hidding relations between two areas (Fig. 9(a)).

A structure H with two hiding edges shall be given:

$$o \xrightarrow{2:t5} o \xleftarrow{1:t1} o$$
$$3{:}\langle hta \rangle \quad 1{:}\langle hta \rangle \quad 2{:}\langle hta \rangle$$

describing the situation pictured in Fig. 9(b).

If we realize the operation represented by edge $1{:}t1$, then one achieves a structure D: $D_A = H_A - \{1\} \cup \{o1, o2, o3, o4\}$, with the edge $2{:}t5$ being replaced by only one edge $3{:}t5(3{:}\langle hta \rangle, o4{:}\langle hta \rangle)$.

Let two visibility operations be given:

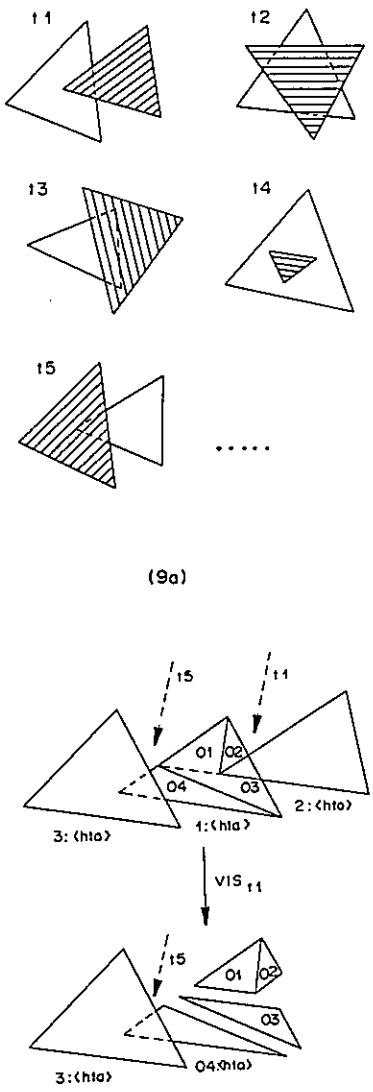


Fig. 9. Example of visibility operation: (a) some hiding situations for two triangle areas, (b) operation VIS_{t1}.

$$D1 = VIS_{t1}[H(i{:}a1 - j{:}b1)],$$

$$D2 = VIS_{t2}[H(w{:}a2 - z{:}b2)].$$

Three types of their dependence occur:

a. At most, the second nodes are the same ($j = z$). If the set of n-edges with terminal hiding labels passing simultaneously the first nodes (i, w) is empty, then these two visibility operations are called INDEPENDENT. Any independent operations can be realized simultaneously.

b. The first nodes are the same. While performing one operation we possibly change the other. These operations can not be performed in parallel.

c. The first node of operation is the same as the second of the other operation. This is the most complicated case, and two operations have to be performed sequentially.

We want the set of visibility operations to be transitive and a VN-structure that is REGULARLY HIDDEN (all compositions of all visibility operations specified in H will result in isomorphic structures).

Definition 11. The VISIBILITY STEP in $H \in$ VNSTR is each composition of all visibility operations specified in H (denoted VIS(H)).

I proofed as follows:

Proposition 12. If $D \in$ VNSTR do not have multiple n-edges, and n-edges do not create cycles in D, then a set of virtual productions with application context (P_{VI}) exists and satisfies:

1. If $H \in \mathrm{VIS}(D)$ then (quasi-visibility)

$$H \in q\mathrm{VIS}(D) = \{H \in V'\mathrm{NSTR} | V' = (V_A, V_E - \mathrm{HID}^T), D(\rightarrow_\omega)^* H\},$$

where "$\rightarrow_\omega$" means virtual direct derivation with P_{VI};

2. If the semantic rule SEM, which determines whether the virtual neighborhoods, is compatible with the semantics of visibility operations then

$$\mathrm{VIS}(D) = \{H \in V'\mathrm{NSTR} | V' = (V_A, V_E - \mathrm{HID}^T), D(\underset{SE}{\rightarrow}_\omega)^* H\},$$

where "$\underset{SE}{\rightarrow}_\omega$" means the virtual direct derivation $\rightarrow_\omega$ in which the values of virtual neighborhoods are determined by SEM.

3.4 *Syntax restrictions*

Definition 13. The system $\langle SP, P_{GL}, P_{VI}, R^S, \overset{GL}{\underset{VI}{\rightarrow}}_\omega \rangle$ is called VIRTUAL nPSG WITH GLUING AND VISIBILITY (denoted $\mathrm{nPSG}^\omega_{GL,VI}$), where

SP = system of virtual noded structural productions;
P_{GL} = finite set of productions with application contexts simulating the gluing steps for $ID \subset V^N$;
P_{VI} = finite set of virtual productions simulating the quasi-visibility step for HID $\subset V^N$;
$R^S = (R^i)_{i \in St}$ = a finite family of started VN-structures over the same set of nodes; $m_A(R^i_A) = V^S_A$ ($i \in St$);
$\overset{GL}{\underset{VI}{\rightarrow}}_\omega$ = composite direct derivation being one of the following steps:
  a. direct derivation "$\rightarrow_\omega$" by productions from P;
  b. the gluing step (GLU) simulated by P_{GL};
  c. the quasi-visibility step (qVIS) simulated by P_{VI}.

The language generated by grammar G from class $\mathrm{nPSG}^\omega_{GL,VI}$ is

$$L(G) = \{H \in V^T\mathrm{NSTR} | \exists R^i \subset R^S, R^i(\overset{GL}{\underset{VI}{\rightarrow}}_\omega)^* H\}.$$

If conditions given in propositions (8) and (12) are satisfied by grammar G, then

$$G \subset \mathrm{nPSG}^\omega.$$

A language of each nPSG$^{\omega}$ grammar over a monotonical production set is deterministic. This means we always can prove whether a given structure belongs to such language. Due to the monotonicity of derived nodes we can establish in a finite number of steps if actual derivation can reach a given structure. $\text{nPSG}^{\omega}_{GL,VI}$ has non-monotonical productions. But restricting the number of gluing- and quasi-visibility steps occurring in each language derivation to be finite, the problem will be deterministic. The OSL-grammar will be some subclass of deterministic $\text{nPSG}^{\omega}_{GL,VI}$ grammar, as specified by syntax restrictions given next.

1. OSL-grammar is a HIERARCHICAL $\text{nPSG}^{\omega}_{GL,VI}$, which means:

(a) SP consists of $SP(0), \ldots, SP(W-1)$ ($W \in \mathcal{N}$):

$$V^N = \bigcup_{I=1}^{W-i} \{V^N(I) \cup V^T(I) \cup V^N(0)\}; \quad V^T = V^T(0);$$

$$V^S = V^S(W-1);$$

$$P = \bigcup_{I=0}^{W-1} P(I).$$

(b) $P_{GL} = \bigcup_{I=0}^{W-1} P_{GL}(I)$; $ID(I) \subset V^N(I)$.

(c) $P_{VI} = \bigcup_{I=0}^{W-1} P_{VI}(I)$; $\text{HID}^T(I) \subset V^N(I)$.

(d) At most one GLU- and qVIS-step occur at each level I.

(e) All productions applied in a direct derivation belong to one subset, $P(I)$.

2. The family of started structures consists of N-isomorphic VN-structures varying by n-edge labeling only. Each n-edge has some label $E \subset V_e$ or a complementary label $\bar{E} \subset V_{\bar{e}}$ and $V_E^S = V_e \cup V_{\bar{e}}$.

3. The derivations at one level are of two general types:

$$\text{ALFA: } G^{I+1}[\rightarrow_{\omega}(I) \cdot \text{GLU}(I) \cdot (\rightarrow_{\omega}(I))^* \cdot \text{qVIS}(I)]G^I$$

(G^I—terminal structure at level I),

Recursive derivations perform approximation of "complex" nodes. If at level I primary nodes exist only, a subtype arises:

$$\text{ALFA0: } G^{I+1}[\rightarrow_{\omega}(I) \cdot \text{GLU}(I) \cdot \text{qVIS}(I)]G^I$$

$$\text{BETA: } G^{I+1}[\rightarrow_{\omega}(I) \cdot \text{GLU}(I) \cdot (\rightarrow_{\omega}(I))^*]G^I.$$

4. The set $P_a(I)$ at level of type ALFA is further specified by:

AL0: On level of type ALFA0 productions from the set $P_{a0}(I)$ can be applied only to level $I+1$, terminal structures. They perform the CONSTRUCT step.

$$P_{a0}(I) = \bigcup_{a \in V_A^T(I+1)} P(a).$$

The right structures of productions from $P(a)$ have a common part J called REPRESENTATIVE of node labeled by "a," which do not contain identity or terminal-gluing labeled edges.

The variable parts are called LOCATION structures and consist only of edges labeled by $ID(I)$ or $\text{HID}^T(I)$.

The same inserting rule is generally associated with each production from $P(a)$. Some of the inserting components can be reduced from the rule for particular production.

ALI: If the set of "complex" node labels at level I is not empty ($V_A^c(I) \neq \emptyset$), then this level is of type ALFA. Besides $P_{a0}(I)$ the set

$$P_{ag}(I) = \left(\bigcup_{a \in V_A^c(I)} P_{ga} \right) \cup \left(\bigcup_{b \in V_A^T(I)} P_{idb} \right)$$

exists, where

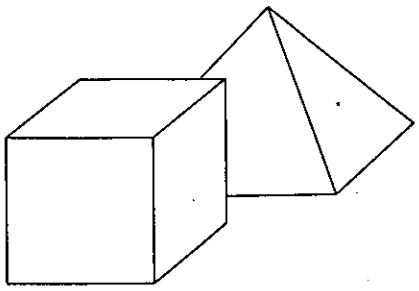
P_{ga} = set of productions performing an APPROXIMATION step for nodes labeled by "a";

P_{idb} = identity productions for level I, terminal nodes.

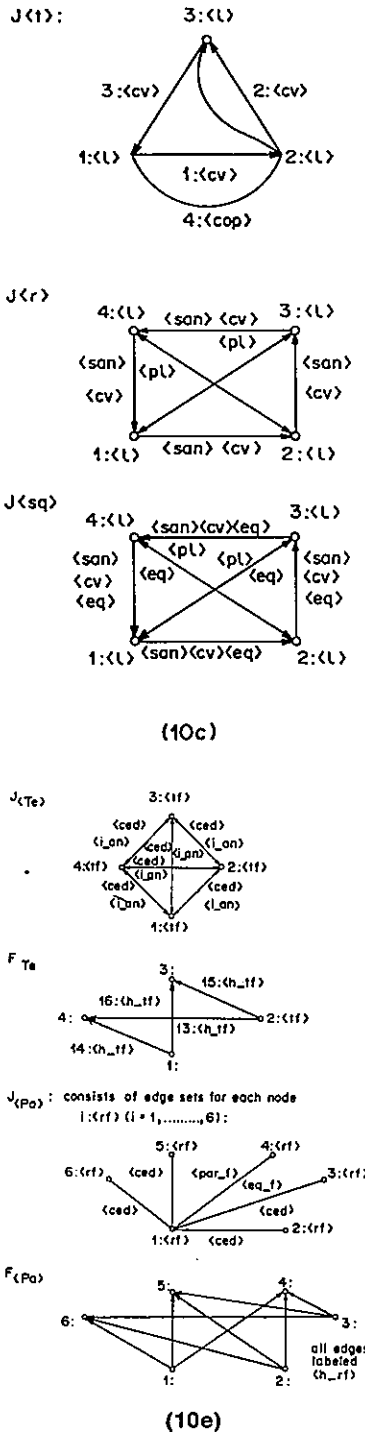
Example 3. Productions from $P_{a0}(I)$ used in CONSTRUCT steps for object (Alfa-example) = {"parallelepiped" + "tetrahedron"} (Fig. 10(a)) can be as follows:

$I = 0$:
$\langle v \rangle$ (vertices), $\langle sl \rangle$ straight line $\in V_A^T(0)$;
$\langle p \rangle$ (start point), $\langle e \rangle$ (end point) $\in V_E(0)$;
$\langle l \rangle$ (line segment) $\in V_A^T(1)$;
$J_{\langle l \rangle}$ = the representative of node "line segment" (Fig. 10(b)).
There is one production:
$p_{\langle l \rangle} := (\{0{:}\langle l \rangle\}, J_{\langle l \rangle}; S)$.

$I = 1$:
$\langle cv \rangle$ (common vertices), $\langle \text{cop} \rangle$ (coplanar), $\langle \text{san} \rangle$ (straight angle),
$\langle eq \rangle$ (equal), $\langle pl \rangle$ (parallel), $\langle \text{nco} \rangle$ (not complete segment) $\in V_E(1)$;
$\langle cv \rangle \in ID(1)$;
$\langle \text{nco} \rangle \in \text{HID}^T(1)$;
Representatives and productions for:
$\langle t \rangle$ (triangle), $\langle r \rangle$ (rectangle), $\langle sq \rangle$ (square) $\in V_A^T(2)$.
$J_{\langle t \rangle}, J_{\langle r \rangle}, J_{\langle sq \rangle}$ (Fig. 10(c));
One production for $\langle t \rangle$ — $p_{\langle t \rangle} := (\{0{:}(t)\}, J_{\langle t \rangle}; S)$.
Two productions for $\langle r \rangle$ — $p^i_{\langle r \rangle} := (\{0{:}\langle r \rangle\}, J_{\langle r \rangle}; S^i)$, where S^1, S^2 varies due to assumption which pair of vertices with segment fragments must be seen (some visibility operations can not occur). One production for $\langle sq \rangle$ — $p_{\langle sq \rangle} := (\{0{:}\langle sq \rangle\}, J_{\langle sq \rangle}; S)$.

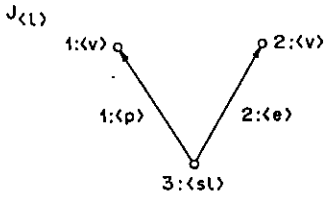


(10a)

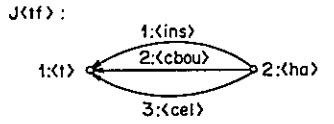


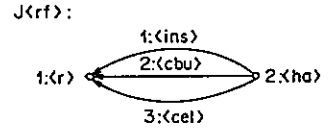
(10c)

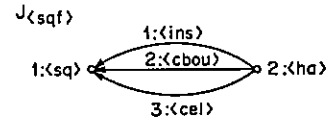
(10e)



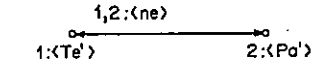
(10b)







(10d)



(10f)

Fig. 10. Example of ALFA-type productions. (a) Object (Alfa-example), (b) Representative on level 0, (c) Representatives on level 1, (d) Representatives on level 2, (e) Representatives and location-structures on level 3, (f) Representative on level 5.

$I = 2$:

$\langle t \rangle$, $\langle r \rangle$, $\langle sq \rangle$, $\langle$hta$\rangle$ (homogenous triangle area) $\in V_A^T(2)$;

$\langle ha \rangle$ (homogenous area) $\in V_A^S(2)$;

$\langle$ins$\rangle$ (inside of), $\langle$cel$\rangle$ (common boundary element), $\langle$cbou$\rangle$ (common boundary) $\in V_E(2)$;

$\langle tf \rangle$ (triangle face), $\langle rf \rangle$ (rectangle face), $\langle$sqf$\rangle$ (square face) $\in V_A^T(3)$;

Only one production for each $\langle tf \rangle$, $\langle rf \rangle$, $\langle$sqf$\rangle$ exists, with representatives like given on Fig. 10(d).

$I = 3$:

$\langle$ced$\rangle$ (common edge), $\langle i\_an \rangle$ (intersection angle = 60°),

$\langle h\_tf \rangle$ (hiding of triangle face), $\langle h\_rf \rangle$ (hiding of rectangle face), $\langle$par$_f \rangle$ (parallelity of faces), $\langle eq\_f \rangle$ (equality of faces) $\in V_E(3)$;

$\langle$ced$\rangle \in ID(3)$; $\langle h\_tf \rangle$, $\langle h\_rf \rangle \in \mathrm{HID}^T(3)$;

$\langle Te \rangle$ (tetrahedron), $\langle Pa \rangle$ (parallelepiped) $\in V_A^T(4)$.

One production for each $\langle Te \rangle$, $\langle Pa \rangle$ (Fig. 10(e)):

$p_{\langle Te \rangle} := (\{0{:}\langle Te \rangle\}, J_{\langle Te \rangle} \cup F_{\langle Te \rangle}; S)$, where

$J_{\langle Te \rangle}$ is the representative and $F_{\langle Te \rangle}$ the location structure (assuming that only three faces are visible).

$p_{\langle Pa \rangle} := (\{0{:}\langle Pa \rangle\}, J_{\langle Pa \rangle} \cup F_{\langle Pa \rangle}; S)$, where

$J_{\langle Pa \rangle}$ is the representative and $F_{\langle Pa \rangle}$ the location structure (assuming that only three faces are visible).

$I = 4$:

There is only relabeling: $\langle Te \rangle \rightarrow \langle Te' \rangle$, $\langle Pa \rangle \rightarrow \langle Pa' \rangle$.

$I = 5$:

$\langle ne \rangle$ (near by) $\in V_E(5)$; $\langle$ex$\rangle$ (object "Alfa-example") $\in V_A^T(6)$.

$J_{\langle ex \rangle}$ (Fig. 10(f)).

5. The set $P_\beta(I)$—productions of BETA-type level I is specified by:

$BE1$: $P_\beta(I) = P_{\beta 0}(I) \cup P_{\beta g}(I)$.

$BE2$: $P_{\beta 0}(I) = \bigcup_{a \in V_A^T(I+1)} p_a$ (This set performs the CONSTRUCT step.)

The right structure of each production p_a either has only level I-terminal labeled nodes or has some of them labeled level I-nonterminal.

$BE3$: Productions from $P_{\beta g}(I)$ perform the CONSTRUCTi step.

Each nonterminal node can be iteratively expanded into itself and one terminal node (expanding productions) or only into terminal one (terminating productions).

Appropriately the edges have an iterative construction.

From a level $I+1$—terminal node only level I—terminal structures from one family of structures can be derived:

For example, $G(k_1, k_2, \ldots, k_l)$, where $k_1, \ldots, k_l$ = number of recursive-nodes-occurrencies.

For each family a minimal structure is if $k_1 = \cdots = k_L = 1$.

I found such sets of productions powerful enough to describe solids with varying structure by holding a parsing algorithm simple.

Example 4. Productions of type BETA define the construction of solid "bottle" (Fig. 11(a)) from volumes.

$I = 4$:

$\langle$cyl$\rangle$ (cylinder), $\langle$ell$\rangle$ (ellipsoid) $\in V_A^T(4)$;

$\langle bd \rangle$ (body) $\in V_A^N(4)$;

$\langle$bot$\rangle$ (bottle) $\in V_A^T(5)$;

$\langle$csym$\rangle$ (common symmetry axis), $\langle eq \rangle$ (equal dimensions), $\langle sm \rangle$ (smaller cross-section), $\langle nt \rangle$ (is not touching), $\langle$tou$\rangle$ (touching by cross-sections) $\in V_E(4)$;

There is only one production for $\langle$bot$\rangle$-

$$p_{\langle \text{bot} \rangle} := (\{0{:}\langle \text{bot} \rangle\}, B^{\langle \text{bot} \rangle}; S) \in P_{\beta 0}(4),$$

where:

$$B_A^{\langle \text{bot} \rangle} = \{1{:}\langle \text{cyl} \rangle, 2{:}\langle \text{cyl} \rangle, 3{:}\langle bd \rangle\},$$

$$B_E^{\langle \text{bot} \rangle} = b_o^k \cup b_o^1 \quad \text{(Fig. 11(b))}.$$

The set $P_{\beta g}(4)$ besides identity productions for level 4—terminal nodes contain one "expanding" production, $p_{\langle \text{bot} \rangle, 1}$, and one "terminating" production $p_{\langle \text{bot} \rangle, T}$:

$p_{\langle \text{bot} \rangle, 1} := (\{0{:}\langle bd \rangle\}, B_1^{\langle \text{bot} \rangle}; S_{b_o} \cup S')$,

$B_{1_A}^{\langle \text{bot} \rangle} = \{1{:}\langle \text{ell} \rangle, 2{:}\langle bd \rangle\}$,

$B_{i_E}^{\langle \text{bot} \rangle}$ (Fig. 11(c)) (in general denoted B_i, where "i" is the identifier of successively derived node $\langle$ell$\rangle$);

S_{b_o} has inserting components which allow to derive n-edge sets:

$$b_{ok}^1 \text{(Fig. 11(d))}, \quad b_1^1 \text{ or } b_2^1 \text{(Fig. 11(e))}$$

and to transfer the previously derived set B_i.

$$p_{\langle \text{bot} \rangle, T} := (\{0{:}\langle db \rangle\}, \{1{:}\langle \text{ell} \rangle\}; S_{b_T} \cup S''),$$

S_{b_T} has components, which allow to transfer the previously derived sets of type b_{ok}^1 and B_i.

The minimal structure of family for $\langle$bot$\rangle$ contains only one ellipsoid (Fig. 11(f)). From this structure the bottom-up parsing on level 4 for node $\langle$bot$\rangle$ will start.

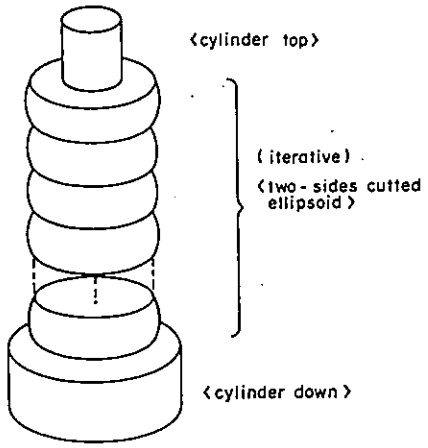
Thus a generative grammar was defined that specifies the syntax of RULES and DERIVATIONS in an OBREC system.

3.5 *Generalization rules*

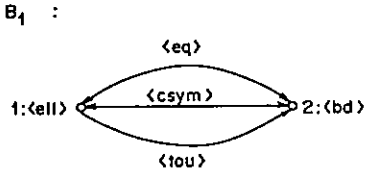
Additional assumptions must be done.

$G1$: For each level of type ALFA, to each set $P_a \subset P_{a0}(I)$ an assistant set of structures KER_a called kernel set is given, satisfying the following:

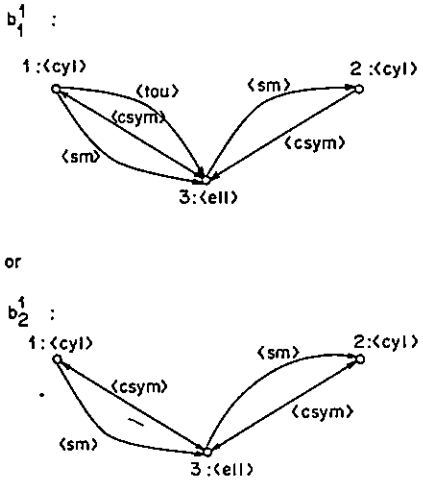
- Each kernel is a substructure of the representative, associated with one production from P_a.



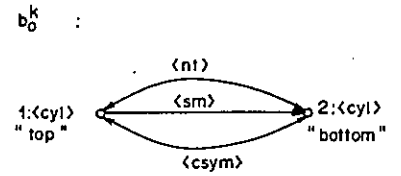
(11a)

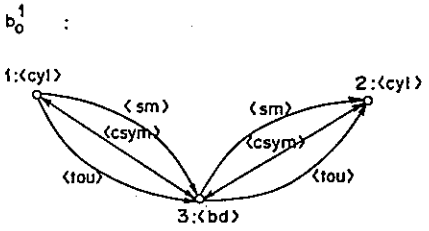


(11c)

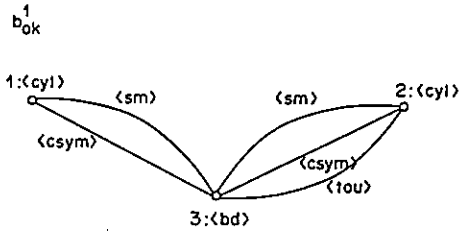


(11e)

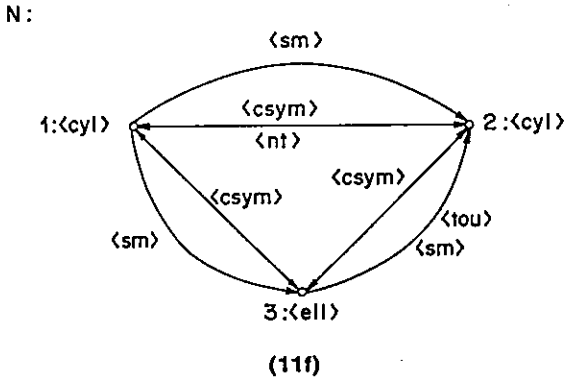




(11b)



(11d)



(11f)

Fig. 11. Example of BETA-type productions. (a) Solid ⟨bottle⟩, (b) Structure $B^{\langle bot \rangle}$, (c) Structure B_1, (d) Structure b^1_{ok}, (e) Structure b^1_1 or b^1_2, (f) Minimal structure N.

- $(d_1, d_2 \in \mathrm{KER}_a)_1([d_1]_\sim = [d_2]_\sim) \Rightarrow ([d_1]_\sim \not\subset [d_2]_\sim)_1([d_2]_\sim \not\subset [d_1]_\sim)$.

The kernels are necessary conditions for recognizing a representative.

Example 5. Kernels associated with productions from Example 3 are:

$I = 0$:

$\mathrm{KER}_{\langle I \rangle} = J_{\langle I \rangle}$ – one kernel;

$I' = 1$:

One kernel for $\langle t \rangle$ represents the situation, where only one side of triangle (Fig. 12(a)) is completely visible.

Two kernels for $\langle r \rangle$ represent the situations, where two vertices with edge-fragments are visible (Fig. 12 (b)).

One kernel for $\langle sq \rangle$ represent the same condition.

$I = 2$:

There is one kernel for each $\langle tf \rangle$, $\langle rf \rangle$, $\langle \mathrm{sqf} \rangle$ (Fig. 12(c)). They allow hypothesizing, if loops of edges with included fragments of homogeneous areas are given.

$I = 3$:

One kernel for each $\langle Te \rangle$, $\langle Pa \rangle$ (Fig. 12(d)). The situation recognized by them is that two (or three) faces are visible.

$I = 5$:

$\mathrm{KER}_{\langle ex \rangle} = J_{\langle ex \rangle}$.

G2: The APPROX-INV step for one "complex" node is a parsing by productions $P_{\beta g}$-like denoted $P_{api}(I)$ (without virtual operators). There must been relationships between sets:

$L_g(a)$—level I—terminal structures derivable from node $a \in V^c_\lambda(I)$ by productions $P_{\alpha g}(I)$,

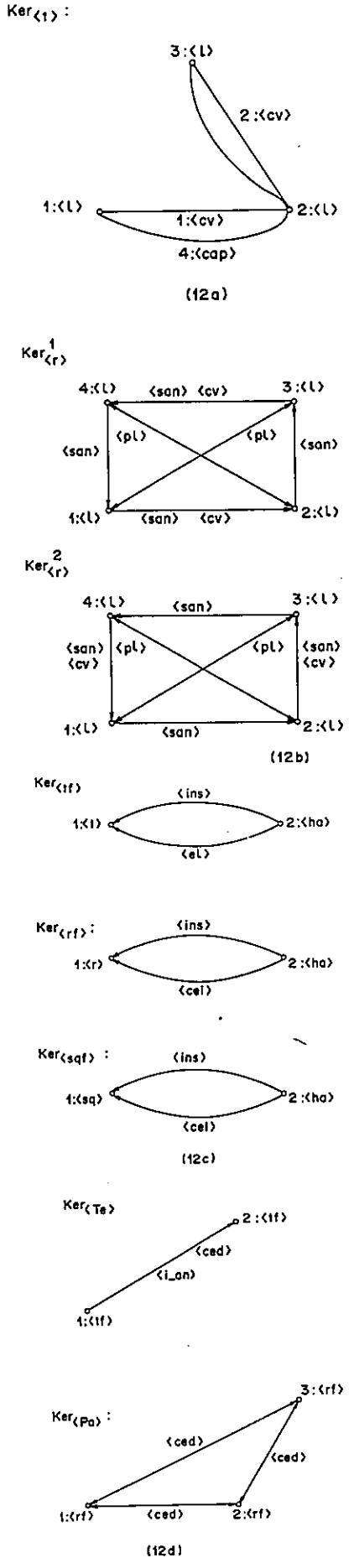


Fig. 12. Kernel structures for object ⟨Alfa-example⟩: (a) for triangle, (b) for rectangle, (c) for faces, (d) for volumes.

$L_{api}(a)$—level I—terminal structures derivable from this node by productions $P_{api}(I)$;

specified as follows:

1. $L_g(a) \subset L_{api}(a)$
2. if (str $\subset L_{api}(a)$) then ($\exists ap \in L_g(a)_1$ str $\subset ap$).

4. SEMANTICS OF OSL

How usual I want the process of meaning evaluation for each language sentence to be depends upon the derivation "history" of the given sentence.

4.1 *Attributed grammar*

I use a many sorted First Order Predicate Language (LS). Its definition follows the standard, so I introduce only the notation here. A signature of language LS consists of sets of identifiers for the following:

- Sorts S
- Functions F
- Relations R
- Constants C
- Special symbols

One starts with introducing variables x_i^s ($i \in \mathcal{N}$, $s \in S$) in order to define the sets of terms—$\mathcal{T}$ and formulas—$\mathcal{F}$.

A substitution for variables is denoted by $O = \{t_1/v_1, \ldots, t_k/v_k\}$, where t_i stands for terms, and v_j for variables ($i = 1, \ldots, k$). A conventional proof system can be given by means of axioms and two deduction rules: modus ponens and generalization.

Then an INTERPRETATION function φ for first-order language and a REALIZATION Z (semantic domain of LS) are introduced, defining the Tarski semantics:

$$(Z, \varphi) \neq \Psi, \text{ where } \Psi \in \mathcal{F}.$$

The nodes of VN-structures should be linked with subsets of sorts from S. A family of sort-subset-pairs

$$H = (H(w; \bar{w})) \quad w, \bar{w} \in S^*$$

is called S-DICTIONARY, and each symbol $h \in H(w; \bar{w})$ is called a S-attributed symbol with synthesized sorts 'w' and derived sorts '$\bar{w}$'.

The S-dictionary induces families of:

S-ATTRIBUTED VARIABLES

$$XH = \{X_l h = (x_1^{s_1}, \ldots, x_l^{s_{k+n}}) Ih \in H(s_1 \cdots s_k; s_{k+1} \cdots s^{k+n}), x_l^{s_l} \in X(i = 1, \ldots, k+n) k, n \geq 0, l \in \mathcal{N}),$$

S-ATTRIBUTED SUBSTITUTIONS

$$\theta X = \{\theta X_l = (t_1/x_1^{s_1}, \ldots, t_{k+n}/x_l^{s_{k+n}}) Ih \in H(s_1 \cdots s_k; s_{k+1} \cdots s_{k+n}), x_l^{s_l} \in Xh \in XH, t_i \in \mathcal{T}(i = 1, \ldots, k+n; k, n \geq 0; l \in \mathcal{N})\}.$$

We restrict our interest only to symbols from $H(w,$

$\lambda)$, $H(\lambda, w)$, $H(w, \bar{w})$ if $w = \bar{w}$ and to constants from $H(\lambda, \lambda)$, where "λ" stands for "null."

Definition 14. A REALIZATION of language $V(LS)$ (an ATTRIBUTED noded parallel structure grammar) is each system

$$\mathcal{A} = \langle G, (\text{sym}, \text{Term}), (RP, m), Z \rangle,$$

where

- G is a nPSG$^{\omega}$ grammar over productions from $P(V)$;
- (sym, Term) is a pair of coincidence functions between VN-structures and elements from LS defined as follows:
  - sym = (syma, syma′, syme),
    syma: $V_A \to H(w, \lambda)$; syma′: $V_A \to H(\lambda, w)$ and (syma(a) = syma′(a) or syma(a) = λ or syma′(a) = λ);
    syme: $V_E \to \{[\Psi]\}$, $\Psi \in \mathcal{F}$.
  - Term = (hA, t_A, t_E),
    hA: $XH \to \mathcal{N}xV_A$, $hA(X_i h) = 1{:}a$ for h = (syma(a), syma′(a));
    t_A: $\mathcal{N}xV_A \to \mathcal{F}$, $t_A(1{:}a) = \Psi_a(X_i h)$, h = syma(a);
    t_E: $\mathcal{N}xV_E \to \mathcal{F}$, $t_E(1{:}e(l_1{:}a_1, \ldots, l_k{:}a_k)) = \mathcal{T}_e(x_1 h_1 \cdots X_k h_k)$, where h_i = syma(a_i), $i = 1, \ldots, k$.
- RP is a finite set of SUBSTITUTION RULES

$$RP = \{rp = (\theta X_1 \cdots \theta X_l;\ OX_{l+1} \cdots OX_{l+r}) \in \theta X^* \times \theta X^*\};$$

  where derived elements of $OX_1 \cdots OX_2$ and synthesized elements of $OX_{L+1} \cdots OX_{L+r}$ are defining and all others are applying, satisfying:
  - All defining elements are of form x/x.
  - All variables occurring in applying elements occur also in defining elements in the same rule.
- m: $2^{P(V)} \to RP$ is a mapping of production-subsets into substitution rules which satisfies the mapping hA.
- Z is a realization of LS.

Due to the attributation, the semantics of a configuration will be given indirectly by the interpretation of LS-components associated with this configuration.

The language defined by the attributed RGS will consists of pairs:

$$(\langle \text{sentence} \rangle, \langle \text{meaning} \rangle).$$

Definition 15. Let Q be an attributed RGS, and $G \subset Q$ is an RGS. Let $h \in L(G)$, K be a configuration in Q whose last structure is h. The MEANING of sentence h is each vector of values ("val") for S-attributed variables associated with start_structure-nodes given by some interpretation φ and $\varphi(K) = \mathbb{1}$.

Two language expressions, ($h1$, $m1$), ($h2$, $m2$), are "semantically equal," if there exist configurations $k1$, $k2$ and interpretations $\varphi 1$, $\varphi 2$, satisfying: $(\varphi 1(k1) = \varphi 2(k2) = \mathbb{1}) \wedge (m1 = m2)$.

4.2 *Semantic restrictions*

In accordance with the syntax, there are some semantic restrictions.

$S1$. for

$$E \in V_e^S(t_E(1{:}E) = \Psi \Rightarrow t_E(1{:}\bar{E}) = \sim\Psi).$$

(For each interpretation at most one start_structure is semantically "true.")

$S2$. for

$$a \in V_A^T \quad \text{and} \quad \text{syma}(a) = h(s_1 \cdots s_k; \lambda)$$

all domains $A^{S_1}, \ldots, A^{S_k}$ must be FINITE.

$S2$ is a necessary condition for the set of language-expressions to be enumerative. This is reality, since the scene is discrete.

$S3$. The mapping m is further specified by the following:

a. If at direct derivations $\to$, $\to_{\omega}$ only monotonical productions are applied, then with each such production a substitution rule from type

$$(\theta X_0; \theta X_1 \cdots \theta X_k) \in \theta X \times \theta X^k$$

is associated.

b. With each pair of productions performing the gluing or quasi-visibility operation (one of which is an erasing production), one substitution rule is associated:

$$(\theta X_1, \theta X_2; \theta X_3) \in \theta X^2 \times \theta X \quad \text{(for gluing)},$$

$$(\theta X_1, \theta X_2; \theta X_3 \cdots \theta X_k) \in \theta X^2 \times \theta X^{k-2} \quad \text{(for visibility)}.$$

Other productions satisfy point a.

$S3$. The S-attributed variables in substitution rules are

- Only synthesized in rules associated with productions from $P_{\beta}(I)$
- Only derived in rules associated with P_{GL}, P_{VI} and $P_{\alpha g}(I)$
- Synthesized and/or derived in rules associated with $P_{\alpha 0}(I)$

5. CONCLUSIONS

In this paper I have studied a subtask of depth-map and intensity-image understanding. Its model in terms of attributed structure grammar was presented. In this way an extension of 2-dimensional pattern recognition models for recognition objects in discrete 3-D space was done. The study of approximation- and approx-inv-steps is still needed.

Some implications for the design of specialized hardware arise. Controlled by a host computer, a multiprocessor facility can help effectively to perform the recognition strategy:

- The direct derivation in nPSG shall be performed in parallel.
- All predicates of LS language can be interpreted by specialized hardware.
- Matching of VN-structures shall be a task for associative processors.

REFERENCES

1. A. R. Hanson and E. M. Riseman (Eds.), *Computer Vision Systems* Academic Press, New York (1978).
2. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* Wiley, New York (1973).
3. J. Kittler, K. S. Fu and L. F. Pau (Eds.), Pattern recognition theory and applications. *Proceedings of the NATO Advanced Study* North-Holland, Dordrecht (1982).
4. K. S. Fu, *Syntactic Methods in Pattern Recognition.* Academic Press, New York (1974).
5. M. G. Thomason, Syntactic/semantic techniques in pattern recognition. A survey. *Int. Journal Computer Information Sciences* **11**, 75–100 (1982).
6. R. S. Michalski, Pattern recognition as rule-guided inductive inference. *IEEE Trans. Pattern Anal. Machine Intelligence* **2**, 349–360 (1980).
7. R. M. Haralick, Using perspective transformations in scene analysis. *Computer Graphics and Image Processing* **13**, 191–221 (1980).
8. Y. Yakimowsky and R. Cunningham, A system for extracting three-dimensional measurements from a stereo-pair of TV camera. *Computer Graphics and Image Processing* **7**, 195–210 (1978).
9. Jarvis, R. A., A perspective on range finding techniques for computer vision. *IEEE Trans. Pattern Anal. Machine Intelligence* **5**, 112–139 (1983).
10. D. Nitzan, A. E. Brain and R. O. Duda, The measurement and use of registered reflectance and range data in scene analysis. *Proceedings of the IEEE* **65**, 206–220 (1977).
11. B. Neumann, Knowledge sources for understanding and describing image sequences. In: W. Wahlster, *GWAI-82. 6-th German Workshop on Artificial Intelligence,* 1–21. Informatik Fachberichte, IfB 58, Springer Vg., Berlin (1982).
12. H. Niemann, Pattern analysis. *Springer Series in Information Sciences* **4**, Springer Vg., Berlin (1981).
13. H. Ehris and H.-J. Kreowski, Parallel graph grammars. In: *Automata Languages Development,* pp. 425–442. A. Lindenmayer and G. Rozenberg (Eds.), North-Holland, Amsterdam (1976).
14. H. Ehris, M. Nagl and G. Rozenberg, Graph-grammars and their application to computer science. *Lecture Notes in Computer Science* **153**. Springer, Berlin (1983).
15. M. Nagl, *Graph-Grammatiken. Theorie, Implementierung, Anwendungen.* Vieweg, Braunschweig (1979) (in German).
16. W. Kasprzak, Model of a three-dimensional object recognition system. Ph.D. dissertation, Technical University of Warsaw, Faculty of Electronic Engineering, Warsaw (1986) (in Polish).