CHAPTER 9
PATTERNS IN MATHEMATICAL SEMANTICS

9.1. Introduction

In this chapter we shall introduce mathematical semantics as the pattern theoretic study of mappings between image algebras and formal languages.

The image algebra will be synthesized using generators that represent relations. This will serve as the semantic counterpart of a formal language. Then the image algebra is studied in terms of similarities, local and global regularity.

The semantic map will be seen to form a category, in the algebraic sense of the term, and we shall examine its morphisms.

We shall also present strategies for constructing semantic maps with special properties related to memory requirements.

Some examples and computer experiments will be given to supplement the analytical treatment.

9.2. Introducing mathematical semantics

2.1.1. Can pattern theory contribute anything to the study of semantics and to the study of how semantics is learned (should be learned) by man (machines)? The word semantics is of fairly recent origin, dating back to the XIX century, but the subject itself goes back to the beginnings of philosophy. Most of the major figures in the history of philosophy devoted some of their thinking to the relation between words, sentences, grammar, and language, on the one hand, with phenomena in the real world on the other.

Such studies have traditionally been carried out by informal means and involved no explicit use of mathematics.

2.1.2. More recently attempts have been made to formalize semantic ideas, which can be seen especially in two disciplines: linguistics and computer science. In *formal* linguistics this seems to have been started at about the same time as when the study of syntax was formalized during the 1960's. The earliest reference that we are aware of is Katz-Fodor (1963), where syntactic structures were transformed into what has become known as K-F trees. The K-F trees are formal constructs attributing meaning to linguistic utterances.

Linguists have continued along this avenue of approach, which has resulted in a large literature. An important idea in this literature is the semantic net which has been applied many times. One has typically taken a subset of a natural language, usually English, and tried to formalize its semantics by a computer program. In this way one would hope that the logical discipline and precision required when writing the program would bring out the basic difficulties clearly.

An important contribution can be found in Woods (1970). The interested reader will find an interesting presentation of this approach in Simmons (1973).

2.1.3. These endeavors overlap to a considerable extent with work done in artificial intelligence, although the emphasis differs. In the latter the goal is often to build a question-answer program for some sufficiently narrow domain of discourse. The well-known work by Winograd (1972) belongs in this group.

The many attempts that have been made in this direction aim at, not just a computer program, sometimes possibly of utilitarian value, but insight and understanding of semantic structures. In spite of skeptical comments to the contrary we believe that these efforts, some of which were mentioned above, have indeed led to an increased understanding.

2.1.4. As far as we know, mathematical formalization has not been employed except in a few publications. One is in Sandewall (1971), where the mathematical tool is predicate calculus.

In 1977 the author together with P. Wegner organized a seminar series in formal semantics at Brown University. During this series the voluminous literature was surveyed, most of it from the linguistic and computer science journals. Formalization in mathematical terms seems to have been attempted only sporadically, and we came across little of mathematical content.

One reason why mathematics has been used so little is probably that no mathematical theory has appeared suitable for the analysis of semantic structures. We believe that pattern theory offers a tool suitable for this purpose. The

present section is a continuation of work begun in Volume II, Section 2.4. It was reported in Grenander (1978b).

In particular we shall attempt to show that mathematical semantics can be expressed in terms of mappings of configuration spaces and image algebras. Such mappings are fundamental to pattern theory, just as morphisms are fundamental in algebra in general.

2.2.1. Our perspective is conformal to that of the early Wittgenstein in his *Tractatus Logico-Philosophicus*, except, of course, that we shall proceed in a mathematically formalized manner. In the next sections we shall remind the reader of Wittgenstein's view of the issues that will concern us here. Some of his aphorisms have been reproduced in an Appendix.

Wittgenstein is often as obscure as he is thought provoking, perhaps intentionally so. When he speaks of "things" for example, it is not clear if these are material objects or, say, sensory data. See Notes A.

2.2.2. The world consists of facts, T1.1-1.12 (this refers to the numbered sections of *Tractatus*). A fact is a collection of things related to each other, T2.0272, 2.031. The things make up the substance of the world, T2.021.

Some facts can be seen to be made up from other facts, others cannot be split up. The latter are the atomic facts.

2.2.3. Let us denote the set of things by T and consider a set O of operations. The operations act upon things and produce simple facts. An operation can operate on just one thing, or two things, and so on. It is a function with, say, n places (or arguments).

When we apply all operators to all combinations of things we get the set S of atomic facts. Wittgenstein probably does not assume that an operator with n places can be applied to any combination of n things. If this is so the operations are partial functions.

Another set U of operations acts upon atomic facts, from S, and results in composite facts. The set F of all such facts is the ontological base for understanding the world.

2.2.4. Of course Wittgenstein did not formalize his thinking in this way, perhaps he would be opposed to *any* formalization attempt. It would be *too* precise, losing the "multi-dimensional" ambiguity.

2.2.5. A picture in *Tractatus* is a model of the world, grouping elements that correspond to things (T.2.13) into structures. A picture is also a fact, T.2.141.

A proposition is made up of names. It is a fact, its elements are related to each other, T.3.14, and it is a picture of a possible grouping of things.

In some sense the structure of the picture should be "congruent" to the real situation it represents. "Congruent" does not mean identical, the correspondence can be more complicated.

This correspondence, if it could be articulated exactly, would associate meaning to propositions. It is likely that Wittgenstein did not have ordinary natural language in mind when he discusses propositions. Perhaps he meant "scientific language", or language as it *ought* to be.

2.2.6. A reader familiar with pattern theory will recognize the similarity between some of its basic concepts

with the thinking in *Tractatus*. The generators correspond to things and operators, T ∪ O. The operators in O have arities, the number of places. Configurations correspond to facts and the connectors allowed in the configuration space correspond to the operators in U. The totality F is the configuration space.

2.2.7. In Sections 3-7 a mathematical formalization of semantics will be given expressed as mappings between two image algebras. The philosophical view of *Tractatus* has influenced this formalization.

In his later years Wittgenstein renounced Tractatus, the work of his youth. We shall have something to learn also from the later Wittgenstein, however, namely about learning semantics.

2.3.1. Our speaker/listener will be immersed in a world of sensory impressions. Based on these sensory data and with the aid of a priori knowledge he, the observer, makes statements or receives statements about the state of the world expressed, we assume, in some formal language L. Since our approach will be abstract, we need not specify whether these statements are just declarative, affirmative, or whether they can be questions, expressing doubt, containing judgments, or be imperative, and so on.

The fact that we shall use examples where the statements look like simple English sentences should not be taken to mean that we are modelling the semantics of English, not even a subset of it. Our goal is to understand certain mathematical phenomena, not linguistic ones. If this can be achieved we hope that the results will in due time have applications to linguistics, but this would be too early to claim at present.

2.3.2. The observer's statements should be correlated to his view of the world. His view will be expressed formally as an image algebra to be examined in Section 3. The image algebra should be mathematically consistent, as will be proved for the one we propose, but it need not be a "true" description of the world.

We are therefore operating on three levels. The "true" world, the formal description of the way the observer views the world, and the linguistic utterances prompted by the view. It is only the relation between the two latter levels that we shall study here.

2.4.1. All natural languages can be ambiguous. This has been pointed out so many times that we need not elaborate this trite fact any further. In context, and with access to linguistic deep structure, ambiguity may perhaps be removed. Whether this is so or not, we shall simply *require* that the grammatical utterances have a unique semantic content.

Most of our attention will then be paid to the study of such semantic maps, their mathematical construction and analysis of their properties, especially of their memory requirements and limitations. This will be done in Sections 6 and 7. In the last sections of this chapter we shall study the learning of semantic maps.

2.4.2. When mathematics is applied to any subject matter one is forced to simplifications, sometimes drastic ones. This is certainly true here; a narrow range of situations will be analyzed in some depth at the cost of introducing specializing assumptions. The abstract treatment is hoped to bring out the logical essence of the problem as clearly as possible. This will avoid vague generalities and bring into the open

hidden assumptions, albeit at the price of restricting the scope of the results.

2.4.3. In order to pinpoint the concepts needed for the mathematical analysis our reasoning, we shall be dialectic, arguing for and against adopting certain notions and assumptions. In this way we have arrived at a formalization that we hope will be useful for our later work.

2.5.1. The abduction machine analyzed in Grenander (1978), Chapter 7, *creates syntactic hypotheses* sequentially, tests them and accepts or rejects them. In a certain well-defined linguistic situation it was proved to yield, ultimately, a set of correct hypotheses.

In an early theorem (see Notes B) the author showed how syntactic abduction can be achieved for languages of a very general type. This theorem is, however, only of theoretical interest since the algorithm would be very slow due to the fact that it is too general; it does not exploit any underlying structure. Another drawback is that the learning would not be incremental. The syntactic abduction machine mentioned seems better suited to the problem.

2.5.2. Is it possible to build an *abduction machine for semantic hypotheses?* We shall show that mathematically this amounts to estimating a relation from a finite set (consisting of productions for L) to the morphisms of a category. As far as we know this mathematical problem has never been studied up till now; it will be done in 9.8.

9.3. Formalization through regular structures

3.1. Any coherent view of the world must be based on some *notion of regularity*. Otherwise it would be without laws and constancies, with nothing permanent to learn, no structure to discover.

The regularity need not be deterministic. On the contrary, many of the phenomena that we encounter in every day life are ruled by *statistical laws* only. Statistical regularity should therefore be allowed. A mathematical consequence of this is that the state space becomes more sophisticated.

3.2. To formalize a view of the world we need a precise notion of regularity. We shall show in the following that *combinatory regularity* (pattern theory) is logically conformal to the ideas of Section 2.2.

3.3. Let us remind the reader that pattern theory is of algebraic nature and based on the idea of an *image algebra*

$$\mathscr{I} = \langle G, S, \mathscr{R}, R \rangle. \tag{3.1}$$

An image algebra is made up of a set G of *generators*, from which *configurations* are formed following the *rule of regularity*, $\mathscr{R}$. The group S of transformations of G onto G, the *similarities*, expresses which generators are similar to each other. The set of regular configurations $\mathscr{C}(\mathscr{R})$, formed according to $\mathscr{R}$, is divided into equivalence classes, the *images*, by means of the equivalence relation R: the *identification* rule. The images form a partial universal algebra $\mathscr{I}$ with respect to certain connection operations.

We now discuss the choice of each component in (3.1) for the purpose of this study.

3.4.1. The generators $g \in G$ shall be thought of as *relations* in a general sense that will become clearer as we go along.

In Section 6 we shall relate the image algebra to language. Formal linguistics is dominated by the finitistic attitude so that it would seem natural to assume that G is finite.

On the other hand, we would like to let the generators carry attributes such as location, orientation, frequency, time, etc. These are usually thought to be continuous in nature so that we would be led to allow G to be infinite.

For the time being we shall choose the first alternative, $\#(G) < \infty$, reserving the possibility of extending the results to infinite generator spaces.

3.4.2. Generators shall carry two sorts of *bonds*, in-bonds and out-bonds, leading us to directed regularity. The *out-arity* shall be finite and, since G is finite, bounded over G

$$\omega_{out}(g) \leq \omega_{max} < +\infty. \tag{3.2}$$

We are less certain about the *in-arities* $\omega_{in}(g)$. After having examined a large number of cases it became clear that generators should be allowed to accept many in-bonds. Whether this number should be bounded or not is less clear. We choose for the moment to make it unbounded

$$\omega_{in}(g) = +\infty, \quad \forall g \in G. \tag{3.3}$$

Note that all generators have in-bonds but not necessarily out-bonds.

The arities as well as the values "in", "out", associated with every bond belong to the bond structure. Sometimes the different out-bonds have different functions so that it will be necessary to indicate this by other *bond structure parameters*, see Chapter 3. This will be done by markers "1", "2", etc. We then rule out the possibility that some markers are equal, at least for now. For the in-bonds no such markers will be used at present; again this may have to be modified when we have learnt more about the use of these regular structures.

3.4.3. To each bond is associated a *bond value* v, taking values in some set B. We suspect that it would be convenient to make these values subsets of G

$$V = 2^G, \tag{3.4}$$

but in the present discussion this will not be done.

For a given generator the bond values associated with out-bonds may differ, expressing their difference in function. The in-bond values, on the other hand, will be assumed to be the same. The rationale behind this assumption is that out-bonds shall express active properties of a generator (relation) that may vary from bond to bond. The in-bonds express passive properties that are constant for all in-bonds of the generator.

We are aware of examples where this assumption will lead to logical inconsistencies. A generator may accept two in-bonds belonging to two generators, that, viewed as unary relations, expresses properties that are not compatible with each other. Recalling the discussion in Section 2, however, this will be allowed: the observer's view of the world need not

be consistent with the "true" state of the world.

3.4.4. To be able to refer to the bonds of a given generator we need *bond coordinates*. Therefore we shall enumerate the out-bonds by $1,2,3,\ldots\omega_{out}(g)$, with the convention that if some of them have already been marked by the bond structure parameters $1,2,\ldots r$, then this numbering will be adhered to for the bond coordinates. In configuration diagrams bond coordinates will sometimes be put inside parentheses when needed for clarity.

Since all the in-bonds carry the same bond value, at least for now, we need not distinguish between them and shall not use any bond coordinates for them.

3.4.5. Consider a generator g with $\omega_{out}(g) = \omega$, so that its (out-) bond coordinates are $1,2,\ldots\omega$. Let

$$\pi : (1,2,\ldots\omega) \to (i_1,i_2,\ldots i_\omega) \tag{3.5}$$

be a permutation of the ω first natural numbers. To each ν, $1 \leq \nu \leq \omega$, correspond bond structure parameters $B^S_\nu(g)$ and bond values $B^V_\nu(g)$. If

$$\begin{cases} B^S_\nu(g) = B^S_{i_\nu}(g) \\ B^V_\nu(g) = B^V_{i_\nu}(g) \ , \end{cases} \tag{3.6}$$

for all ν, the renumbering π does not affect the connectivity properties of g. The set of all such permutations π forms a subgroup $\pi(g)$ of the symmetric group over ω objects: the *symmetry group* of g.

In the special case when all out-bonds of g carry distinct markers $1,2,\ldots\omega$ the symmetry group consists of the identity element.

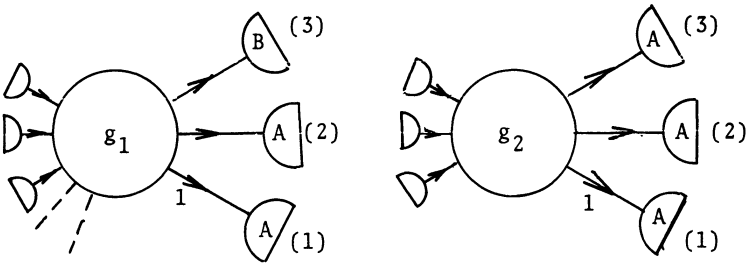


Figure 3.1

In Figure 3.1 the symmetry group $\pi(g_1)$ consists of the identity if $A \neq B$, but $\pi(g_2)$ is of order 2 allowing (2) and (3) to be exchanged without changing the connectivity properties of g_2. Note the bond structure marker "1" at the bottom out-bond in the diagram.

3.4.6. Bonds shall take values in sets $\mathscr{B}_\nu \subset \mathscr{B}$, $\nu \geq 0$. Any generator shall have one and the same in-bond value from some $\mathscr{B}_\nu$ and then its out-bonds, if there are any, shall be from $\mathscr{B}_{\nu-1}$. If $\nu \geq 1$ the generator shall have at least one out-bond. The value of ν expresses the *level of abstraction* of the generator, $g, \ell(g) = \nu$ in a way that will become clear as we go along.

We have one partition of G into sets G_k^ω where $k = \omega_{out}(g)$. Here ω is just a label for "arity", not a variable superscript. Another partition is induced by the level of abstraction into sets

$$G_\nu^\ell = \{g \mid \ell(g) = \nu\}; \quad \nu = 0,1,\ldots ; \tag{3.7}$$

with ℓ a label for "level". We shall refer to generators from these classes as follows:

$$\begin{cases} g \in G_0^{\ell} & \text{as "objects"} \\ g \in G_1^{\ell} & \text{as "properties"} \\ g \in G_2^{\ell} & \text{as "second level relations"} \\ g \in G_3^{\ell} & \text{as "third level relations"} \\ \ldots & \end{cases} \tag{3.8}$$

To each g is associated a number, the level of abstraction, $\ell = \ell(g) = \nu$ denoting the number of the set family $\mathscr{B}_\nu$ to which the in-bonds belong.

Combining all the elements with the same out-arity we get, as mentioned above, a partition into the sets

$$G_\mu^{\omega} = \{g | \omega_{out}(g) = \mu\}. \tag{3.9}$$

Lemma 1. $G_0^{\ell} = G_0^{\omega}$; *objects, and only objects, have out-arity* 0.

Proof: If $g \in G_0^{\ell}$ its in-bond values are in $\mathscr{B}_0$. Since $\mathscr{B}_0$ has no predecessor to which the out-bond values should belong, g can have no out-bonds, so that $\omega_{out}(g) = 0$, $g \in G_0^{\omega}$, and $G_0^{\ell} \subseteq G_0^{\omega}$.

On the other hand, if $g \in G_0^{\omega}$, so that it has no out-bonds, then it cannot have in-bonds with values from any $\mathscr{B}_\nu$, $\nu \geq 1$, see above. Hence $g \in G_0^{\ell}$ which implies $G_0^{\omega} \subseteq G_0^{\ell}$.

Q.E.D.

3.5.1. The *similarities* will be chosen as the set S of all permutations $s: G \to G$ leaving bonds, i.e. bond structure and bond values, unchanged

$$B(sg) = B(g), \quad \forall g \in G. \tag{3.10}$$

It is immediately clear that the permutations s satisfying (3.10) form a group, the similarity group.

Since any s leaves the bond structure invariant, $B^s(sg) = B^s(g)$, it follows that our definition of S is correct, see Volume I, p. 9, except that (ii)(ibid) cannot yet be verified since the generator index has not been defined so far.

3.5.2. Since the present S leaves invariant, not only the bond structure as all similarities do, but also the bond values, it follows that the classification of any g in terms of the set families $\mathscr{B}_\nu$ is also S-invariant. A consequence is that *the level of abstraction is S-invariant*

$$\ell(g) = \ell(sg): \quad \forall g \in G, \quad \forall s \in S. \tag{3.11}$$

3.5.3. We now define a *generator index* class as the set of all g's with the same $B(g)$.

Lemma 2. *This partition is the finest partition by any generator index.*

Proof: If g_1 and g_2 both belong to the same α-class we have $B(g_1) = B(g_2)$. Appealing to (3.10) we see that $B(sg_1) = B(sg_2)$, $\forall s \in S$, which implies that the α-classes are invariant, $\alpha(sg_1) = \alpha(sg_2)$, and hence that α is a legitimate generator index corresponding to the similarity group, see Volume I, Chapter 1, Definition 1.1, (ii).

On the other hand, if α' is some other generator index and $\alpha(g_1) = \alpha(g_2)$ then by definition $B(g_1) = B(g_2)$. The permutation s_o of G that only permutes g_1 with g_2 is therefore a similarity; see (3.10). But all generator indices must be S-invariant so that $\alpha'(g_1) = \alpha'(s_o g_2) = \alpha'(g_2)$ and g_1, g_2 belong to the same α'-class. This shows that α-classes are contained in α'-classes. Q.E.D.

Note that generators with the same index are of the same level of abstraction, since if two generators have the same index α, then they have the same in-bond values. These values then belong to the same set family $\mathscr{B}_\nu$, which leads to the same level of abstraction.

3.5.4. Our choice of generator index could be criticized in that it is too narrow: in order that $\alpha(g_1) = \alpha(g_2)$ hold we must have exactly the same bond structure and bond values for g_1 and g_2. When we exemplify our construction by concrete image algebras this will lead to a classification of generators into very small classes, perhaps too small to be natural. Some modification may be needed as we go along.

3.6.1. We now come to the *rules $\mathscr{R}$ of combinatory regularity*

$$\mathscr{R} = \langle \rho, \Sigma \rangle \tag{3.12}$$

with some bond relation ρ, local regularity, and connection type Σ, global regularity. In accordance with the discussion in Section 2 we want our configurations to consist of relations combined together into a "formula". In order that the formula be "computable" we must choose ρ so that all the connections that are allowed by ρ make sense.

3.6.2. At first it seemed reasonable that the *bond relation* ρ ought to be chosen as INCLUSION. If we think of the generators as logical operators with domains and ranges we are led to operator configurations, see Volume I, Chapter 2, Case 7.1, where INCLUSION was the natural choice.

After examining a number of special cases we have concluded, however, that the more restrictive relation ρ = EQUAL suffices for the present purpose; we choose this

definition for the rest of this chapter.

3.6.3. It is clear that EQUAL is a legitimate bond relation for the similarity group chosen. Indeed, if g_1 connects to g_2 via the bond-values β_1 and β_2, then β_1 must equal β_2. Applying the same similarity s to both g_1 and g_2 will not change the bond values. Hence sg_1 can connect to sg_2 via the same bonds, which shows that EQUAL is legitimate; see Volume I, Chapter 2, p. 27.

3.6.4. This choice of ρ has implications for the levels of abstractions of connected generators.

Lemma 3. *If a generator* g_1 *is connected by an out-bond to an in-bond of* g_2 *then*

$$\ell(g_1) = \ell(g_2) + 1. \tag{3.13}$$

Proof: See Figure 3.2 where the $(k)^{th}$ out-bond of g_1 is connected to an in-bond of g_2. The corresponding bond-values are denoted by β_{1k} and β_{in} respectively. If g_2 is of abstraction level $\ell = \ell(g_2)$ it follows that $\beta_{in} \in \mathscr{B}_\ell$. But ρ requires, in order that the connection be regular, that $\beta_{1k} = \beta_{in}$ so that β_{1k} is also in $\mathscr{B}_\ell$. Then the in-bond value of g_1 must be in the set family $\mathscr{B}_{\ell+1}$ so that $\ell(g_1) = \ell+1$. Q.E.D.

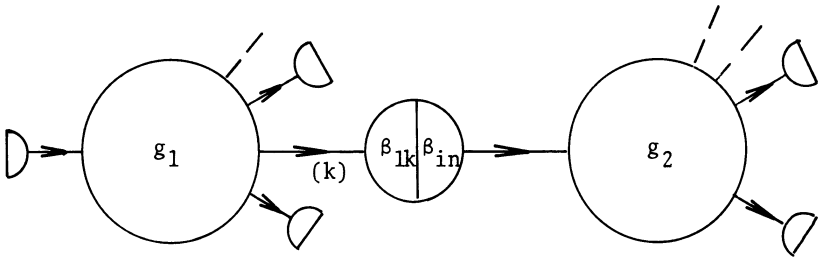


Figure 3.2

Lemma 4. *The generators in any regular configuration* c *have* POSET *structure.*

Proof: Consider a connected component of c with generators $g_1, g_2, \ldots g_n$. All connections go from some level ℓ to some level ℓ-1. Defining $g_i \succ g_j$ if there is a connected chain

$$g_i \to g_{i_1} \to g_{i_2} \to \ldots \to g_j \tag{3.14}$$

it is clear that

$$\ell(g_i) = \ell(g_{i_1})+1 = \ell(g_{i_2})+2 = \ldots \tag{3.15}$$

so that loops cannot occur. It follows easily that "$\succ$" satisfies the postulates of a partial order. Q.E.D.

Generators belonging to two connected components that are not connected to each other, are not ordered with respect to each other. Generators belonging to a connected component are not ordered with respect to each other if they are of the same level of abstraction. Even if they are of different levels it can happen that they are not comparable via "$\succ$".

3.7.1. We are dealing with symmetric regularity: out-bonds can only connect to in-bonds. In this context only finite configurations will occur. The main restriction on Σ will be (in addition to POSET structure as shown in Lemma 4)

$$\Sigma\text{: all out-bonds must be connected.} \tag{3.16}$$

The reason for adopting (3.16) is that we view the out-bonds as active; the logical operator represented by a generator does not make sense unless its arguments are given.

This defines the configuration space in which we will be operating from now on

$$\mathscr{L}(\mathscr{R}) = <G,S,\mathscr{R}>. \tag{3.17}$$

3.7.2. It may be remarked that *this connection type is not monotonic:* if we open some of the bonds or delete some of the generators (and their bonds) from a regular configuration the resulting configuration is not always regular. The reason for this is that we may have opened up an out-bond belonging to the subconfiguration, and this violates (3.16).

Nevertheless we shall have occasion in what follows to deal with such $\mathscr{R}$-irregular subconfigurations. To get this configuration space we apply the functor $\mathscr{MTX}$ to our configuration space

$$\bar{\mathscr{L}}(\mathscr{R}) = \mathscr{MTXL}(\mathscr{R}) \tag{3.18}$$

see Section 3.5. In $\bar{\mathscr{L}}(\mathscr{R})$ all closed bonds satisfy ρ but out-bonds may be left open.

3.7.3. Just as we need coordinates for a generator to be able to refer unambiguously to its bonds, it is convenient to have some way of numbering the generators in a configuration. A configuration will therefore be described here, as several times before, as an indexed set $\{g_i;\ i = 1,2,\ldots n\}$ of generators, each of which has out-bonds with absolute coordinates $(i,1),(i,2),\ldots(i,0_i)$, with $0_i = \omega_{out}(g_i)$; $i = 1,2,\ldots n$. The in-bonds of g_i will have the coordinates $(i,1),(i,2),(i,3),\ldots$. When referring to a bond (i,k) we must also specify whether it is an in- or out-bond.

Such *configuration coordinates* were discussed, but in a general setting, in Section 3.2.

Strictly speaking a configuration is not entirely specified unless expressed via a system of configuration coordinates,

see Notes A. Therefore two configurations c, with generators g_i; $i = 1,2,\ldots,n$; and c', with generators g'_i; $i = 1,2,\ldots,n'$; and with bonds denoted as described, are identical from the functional point of view if and only if

$$\begin{cases} \text{(i)} & n = n' \\ \text{(ii)} & g_i = g'_i;\ i = 1,2,\ldots n \\ \text{(iii)} & \text{bonds connected in c should have their homologues in c' connected, and vice versa.} \end{cases} \tag{3.19}$$

Note that (ii) implies that $B(g_i) = B(g'_i)$ with homologue bonds given by the coordinate system.

More about this in Section 3.8 below when identification is introduced via R.

3.7.4. The cardinality of $\mathscr{L}(\mathscr{R})$ can never be more than denumerable, since we can enumerate $\mathscr{L}(\mathscr{R})$ by first a finite number of configurations in $\mathscr{L}_1(\mathscr{R})$, monatomic ones, then a finite number in $\mathscr{L}_2(\mathscr{R})$, biatomic ones, and so on.

If we exclude the trivial case when $G_o^\omega = \phi$, as will always be done, we can never have $\text{card}[\mathscr{L}(\mathscr{R})] < \infty$. Indeed if $g \in G_o^\omega$ then

$$c = \phi(g,g,\ldots g) \tag{3.20}$$
$$\text{n times}$$

is regular for any n. In (3.20) ϕ denotes the empty connector that does not close any bonds. That $c \in \mathscr{L}(\mathscr{R})$ follows from the fact that all out-bonds in c are connected (there are not any) and ρ holds trivially since no bonds are closed. Hence $\text{card}[\mathscr{L}(\mathscr{R})]$ = denumerably infinite.

3.7.5. The generators in $G_o^\omega = G_o^\ell$, the *objects* (see (3.8)), play a dominant role in regular configurations.

Lemma 5. *All regular non-empty configurations contain objects.*

Proof: Consider an arbitrary $g \in c$ and let ℓ be its level of abstraction. If $\ell = 0$ then g is an object and the assertion holds.

If $\ell \geq 1$ then it has out-bonds in $\mathscr{B}_{\ell-1}$ and Σ requires that they connect to some generator g' of level $\ell-1$. Either $\ell-1 = 0$ so that g' is an object, or we can repeat the argument; eventually we will arrive at some object in the configuration. Q.E.D.

Remark 1. In the monotonic extension $\overline{\mathscr{G}}(\mathscr{R})$ any monatomic configuration is allowed; the level of its generator can then be positive so that configurations consisting entirely of generators more abstract than objects can occur in $\overline{\mathscr{G}}(\mathscr{R})$.

Remark 2. A warning is motivated. "Object" need not represent an object in some material world. As usual, caution is required when mathematical entities are related to concepts used in common sense parlance.

A direct consequence of Lemma 5 is that the only monatomic configuration in $\mathscr{G}(\mathscr{R})$ consist of an object.

3.7.6. The *prime configurations* in $\mathscr{G}(\mathscr{R})$ are easy to characterize.

Lemma 6. *A configuration* $c \in \mathscr{G}(\mathscr{R})$ *is prime if and only if it is connected.*

Proof: If c is not connected it can be viewed as the ϕ-connection of two non-empty and regular configurations c' and $c'' \in \mathscr{G}(\mathscr{R})$. This follows immediately from the fact that the connected components of any c are regular: they satisfy Σ, since all out-bonds are connected, and ρ holds

trivially in them. But if $c = \phi(c',c'')$, c' and $c'' \in \mathscr{L}(\mathscr{R})$ not empty, then c is *composite*, not prime.

On the other hand if c is connected it cannot be expressed as $\sigma(c',c'')$ with both c' and c'' non-empty and regular. Indeed, neither c' nor c'' can have any open out-bonds (this would violate the connection type Σ). But then $\sigma = \phi$ which implies that c' and c'' are not connected to each other, against the assumption. Q.E.D.

It is different for the notion *reducible/irreducible*, see Notes B. A configuration is called reducible if it has some regular proper subconfiguration; otherwise it is irreducible.

Lemma 7. *A regular configuration* c *is irreducible if and only if it is monatomic and consists of a single object.*

Proof: The "if" part is obvious. On the other hand if it is not just an isolated object we can select one of its objects, since Lemma 6 assures that it has at least one. Take the subconfiguration consisting of this object in isolation. It is regular, implying that c is reducible. Q.E.D.

Lemma 7 says that in the present $\mathscr{L}(\mathscr{R})$ we have only trivial irreducible configurations. This is an atypical situation in pattern theory.

3.7.7. What are the "simplest" regular configurations? Fixing a generator g, let us demand that the configuration contain g but has no (proper) regular subconfiguration also containing g. Such a configuration will be said to be a *simple* g-configuration.

Lemma 8. *A configuration* $c \in \mathscr{L}(\mathscr{R})$ *is a simple* g_1*-configuration if and only if* a) g_1 *is the unique solution in* c *of*

$$\ell(g_1) = \max_{g \in c} \ell(g) \triangleq \ell(c) \tag{3.21}$$

and, b) *if all other* $g_i \in c$ *satisfy* $g_i \prec g_1$.

Proof: Assume that c is a simple g_1-configuration with $\ell(c) = \ell$ and that it has another generator g_2 with $\ell(g_2) = \ell$. No descending chain (see 3.6.2) can lead from g_1 to g_2, so that we can delete g_2 with its bonds from c without leaving any out-bonds open. Hence a) holds.

Now assume that c has some generator g_i which is not subordinated to g_1 as in b). Then we can remove g_i with its bonds without leaving any out-bonds open: b) holds.

On the other hand, if c is a regular configuration for which a) and b) hold it cannot be reduced, still keeping g in it. To see that this is so, say that the reduced configuration leaves out g_i from c. Since there is a descending chain from g_1 to g_i some out-bond will be left open, when we delete g_i, unless the entire chain is deleted. But then g_1 is not in the reduced configuration, so that the condition is both necessary and sufficient. Q.E.D.

Lemma 8 tells us that the simple g-configurations have the typical appearance of Figure 3.3, where $\ell(c) = \ell(g) = 3$ and c contains the two objects g_6 and g_7.

How many simple g-configurations are there? With $N = \#(G)$ we get the crude upper bound

$$N \times N^{\omega_{max}} \times N^{\omega_{max}^2} \times \ldots \times N^{\omega_{max}^{\ell}}. \tag{3.22}$$

If $\omega_{max} > 1$ this gives us the bound

$$\frac{\omega_{max}^{\ell+1}-1}{N^{\omega_{max}-1}}. \tag{3.23}$$

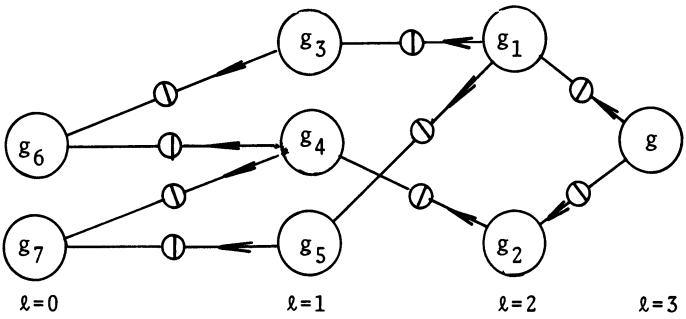


Figure 3.3

The number of simple g-configurations is certainly finite, but it can be extremely large if the abstraction level is big. This will have serious consequences for the ability to learn semantics later on.

3.7.8. Consider a configuration $c \in \mathscr{L}(\mathscr{R})$ with a subconfiguration $c_1 \in \mathscr{L}(\mathscr{R})$. Among the out-bonds of c_1 there may be some that are open. Close these bonds by adding the appropriate generators of c, close the new out-bonds left open, and continue like this. Since c is finite the process will end with some regular subconfiguration c_1^*. In extreme cases $c_1^* = c_1$ or c itself. We shall call c_1^* the *minimal extension* of c_1 in c. The process can be shown to lead to a unique result.

Lemma 9. *For a regular configuration containing the generator* g *the minimal extension* g^* *of the monatomic subconfiguration* $\{g\} \in \mathscr{L}(\mathscr{R})$ *in* c *is a simple* g*-configuration.*

Proof: The regular configuration g^* can have no (proper) regular subconfiguration containing g since if c' were one

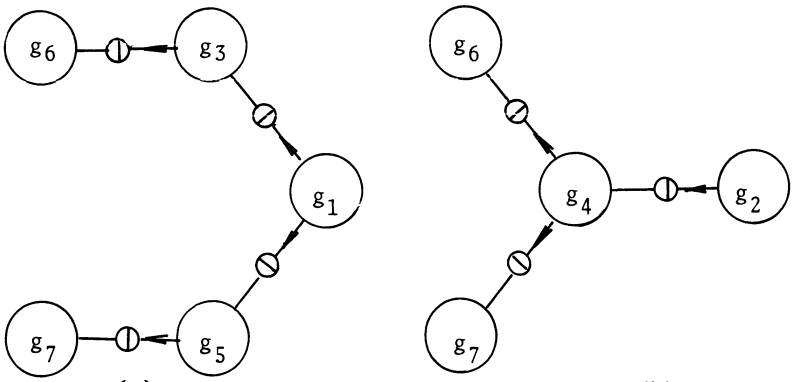


Figure 3.4

such subconfiguration all the out-bonds of g must be closed, just as the out-bond from generators connected to the out-bonds of g, and so on. But then the above procedure gives $c' = g^*$, which proves the assertion. Q.E.D.

As an example, with c as in Figure 3.3, we get g_1^* as in Figure 3.4(a) and g_2^* as in (b).

Remark 1. It is clear that the minimal extension is a closure operation in the limited sense (for a fixed c)

$$\begin{cases} c_1 \subseteq c_1^* \\ (c_1^*)^* = c_1 \end{cases} \tag{3.24}$$

where the inclusion relation denotes the relation configuration-subconfiguration, not just inclusion of sets of generators. For fixed c the minimal extension maps a subset of $\mathscr{C}(\mathscr{R})$ into $\mathscr{C}(\mathscr{R})$.

Remark 2. The operation is also monotonic in the sense that, for fixed $c \in \mathscr{C}(\mathscr{R})$, $c_1 \subseteq c_2 \subseteq c$ implies $c_1^* \subseteq c_2^* \subseteq c$.

3.7.9. As in most pattern theory the *homomorphisms between configuration spaces* play an important role, see

Grenander (1977d). In the present case this is certainly true for $\overline{\mathscr{C}}(\mathscr{R})$ spaces. For $\mathscr{C}(\mathscr{R})$ they are less interesting since all its regular configurations lack open out-bonds. Therefore they can only be connected by the empty connector $\sigma = \phi$, meaning just disjoint union.

Consider now two configuration spaces of the type studied above with

$$\begin{cases} \mathscr{C}_1 = \langle G_1, S_1, \mathscr{R} \rangle \\ \mathscr{C}_2 = G_2, S_2, \mathscr{R} \rangle . \end{cases} \tag{3.25}$$

and with a surjective generator map $\mu: G_1 \to G_2$ preserving bonds $B(\mu g) = B(g)$ and where G_1 and G_2 have the "same" set families, in the sense that $\mu \mathscr{B}_\ell^1 = \mathscr{B}^2$.

Extend the definition of μ to $h: \mathscr{C}_1 \to \mathscr{C}_2$ by putting hc, $c \in \mathscr{C}_1$, equal to the configuration with same connection but where each g_i in c is replaced by μg_i. We then have

Lemma 10. *The configuration map* $h: \mathscr{C}_1 \to \mathscr{C}_2$ *is a homomorphism in the sense of Grenander* (1977d).

Proof: First recall how μ sets up a correspondence between the two similarity groups S_1 and S_2. Each similarity group consists of permutations leaving bonds invariant. But μ preserves bonds, so that the two groups are isomorphic, $S_1 \cong S_2$, although μ need not be bijective. Actually, both of them are the (full) symmetric group of the generator index classes in G_1 and G_2 respectively and these are bijectively related since μ is surjective.

To prove Lemma 10 let $c \in \mathscr{C}_1$ and consider sc, $s \in S_1$. To calculate $h(sc)$ we first have to permute the generators appearing in c according to the similarity c, and then

replace each generator g by μg. But this leads to the same result as if we first replaced each generator by its μ-map value and then permuted them by the isomorphic permutation in S_2: the two operations commute. Recall that the similarities just permute index classes that are defined in terms of equal bonds, and that bonds are preserved by our map.

Finally if $c = \sigma(c_1,c_2)$, with $c,c_1,c_2 \in \mathscr{C}_1$, calculate hc_1 and hc_2 and combine them by the *same* connector σ as before. This is possible since bond values are preserved and ρ is EQUAL; the connection type offers no restriction in the present case since σ is not changed. But $\sigma(hc_1,hc_2)$ will then have exactly the connection of $\sigma(c_1,c_2)$. Its generators have been exchanged according to the generator map μ. Hence $\sigma(hc_1,hc_2) = hc$ as required for h to be a homomorphism.

Q.E.D.

Remark 1. In order that the conclusions of Lemma 10 hold it is not necessary to require that $B(\mu g) = B(g)$. It suffices to ask that a) the bond structure remains the same, $B^S(\mu g) = B^S(g)$, and b) that if $\beta_i(g_1) = \beta_j(g_2)$ then $\beta_i(\mu g_1) = \beta_j(\mu g_2)$.

Remark 2. Also the bond values just mentioned need of course not be exactly the same in the general case when ρ is not EQUAL. It is enough if there exists a map $\beta \to \beta'$ between the two bond value sets for G_1 and G_2 respectively such that the relation $\beta_1 \rho \beta_2$ implies $\beta_1' \rho \beta_2'$. Here β_1 and β_2 refer to g_1 and g_2, while β_1' and β_2' refer to the homologue bonds of μg_1 and μg_2.

This simple fact will be useful later on.

One particular case of some interest later on is when in each index class of G_1 we single out one element g_α, a

prototype, and define μ as $g \to g_\alpha$ if g belongs to G_α. It is easy to show that the conditions of Lemma 10 are satisfied and hence this generator map leads to a homomorphism between the two configuration spaces.

3.8.1. We are now ready to introduce the image algebra. The configurations play the role of "formulas," here satisfying the particular rules $\mathscr{R}$ of combinatory regularity that we have discussed in Section 3.6. The images, on the other hand, express the "function" of these formulas.

In the present case the identification rule R (see Volume I, Section 3.1) will be chosen to make the functions *coordinate free* - the choice of coordinate system for describing configurations should be irrelevant; see Notes C.

To formalize this, consider two regular configurations c and c' with n generators $g_1, g_2, \ldots g_n$ in c and n' generators $g'_1, g'_2, \ldots g'_{n'}$ in c' respectively. The bond coordinates will be denoted by

$$\begin{cases} (k,j), & j=1,2,\ldots n_k \text{ for } g_k;\ k=1,2,\ldots n \\ (k,j), & j=1,2,\ldots n'_k \text{ for } g'_k;\ k=1,2,\ldots n' \ . \end{cases} \tag{3.26}$$

We shall let R identify c and c' if and only if (a) $n = n'$, (b) there exists a permutation $(1,2,\ldots n) \to (i_1, i_2, \ldots i_n)$ such that $g_k = g'_{i_k}$, $k = 1,2,\ldots n$, (c) for each k we have $n_k = n'_{i_k}$, (d) for each k there exists a permutation $(1,2,\ldots n_k) \to (j_1, j_2, \ldots j_{n_k})$ such that the ℓ^{th} bond of g_k equals in structure and value the $j_\ell{}^{\text{th}}$ bond of g_{i_k}, and (e) bonds corresponding to each other are connected/not connected in the same way.

Lemma 11. *This* R *is an identification rule for* $\mathscr{C}(\mathscr{R})$ *and for* $\overline{\mathscr{C}}(\mathscr{R})$.

Proof: That R is an equivalence is obvious. Also, if c and c' are regular, and if they are R-equivalent, cRc', then c and c' have the same unconnected bonds, related by a permutation. These are the external bonds, same for c and c'. We shall then in the following assume that the coordinates have been permuted, if necessary, so that the external bonds are the same for each coordinate. Further, if again cRc', then if we apply the same similarity s to c and to c' we can relate sc and its bonds to sc' and its bonds by the same permutation as for c and c'. Hence (sc)R(sc'). Finally, if $c_1 R c_1'$ and $c_2 R c_2'$, where all four configurations are regular, then c_1 and c_1' have the same external bonds, related by some permutation, and similarly for c_1 and c_2'. Connect c_1 and c_2 into some regular configuration $c = \sigma(c_1, c_2)$. We now connect c_1' and c_2' by the same σ expressed in the coordinate system mentioned above. It follows that $c' = \sigma(c_1', c_2')$ is also regular, since bonds connected in c correspond to bonds connected in c', and vice versa. The bond relation is therefore satisfied for all closed bonds in c'. The connection of c' is also in Σ. But the new c', now known to be regular, consists of the same generators as c, and with the same connections, related by permutations as described. Hence cRc' which shows that R has all the properties of an identification rule. Q.E.D.

Combining Lemma 11 with the properties shown for we have proved

Theorem 1. *With* $\mathscr{L}(\mathscr{R})$ *and* R *as given above* $\mathscr{T} = \langle \mathscr{L}(\mathscr{R}), R \rangle$ *is an image algebra and so is* $\overline{\mathscr{T}} = \langle \overline{\mathscr{L}(\mathscr{R})}, R \rangle$.

In the following all perceptions of the world will be expressed in image algebras of this form.

3.8.2. Any S-invariant class of images forms a *pattern*. Among these are those generated by a *template* I_o

$$P(I_o) = \{sI_o \mid \forall s \in S\} \subseteq \mathscr{I}. \tag{3.27}$$

Two patterns $P(I_1)$ and $P(I_2)$ of the form in (3.27) are either identical or disjoint as subsets of $\mathscr{I}$.

Two distinct images in a pattern describe different perceptions of the world, since otherwise they would be identified by R, but they have the same *logical structure*. That this is so follows from the fact that they are made up of generators, see Notes D, say g for I_1, and then sg for I_2, that have the same generator index and play the same logical, but not substantial, role.

9.4. Two special image algebras

4.1. The construction of the image algebra in the last section is based on simple pattern theoretic ideas. Nevertheless, it takes some experience of manipulating configuration spaces and their images before one becomes familiar with such structures. To facilitate this for the reader we now present two fairly simple examples, the first completely abstract, the second with an intuitive interpretation.

They may appear as quite different from each other but, as we shall show in Section 4.4, they are closely related to each other.

4.2.1. Let G_1 consist of the 12 abstract generators in Table 4.1. As identifiers we have chosen simply the letters of the alphabet and as bond values Roman numerals.

TABLE 4.1

GENERATORS FOR $\mathscr{I}_1$

number	identifier	level	β_{in}	ω_{out}	β_{out1}	β_{out2}	β_{out3}	β_{out4}	β_{out5}
1	a	3	-	2	I	I	-	-	-
2	b	3	-	1	I	-	-	-	-
3	c	2	I	1	II	-	-	-	-
4	d	2	III	1	II	-	-	-	-
5	e	2	III	1	II	-	-	-	-
6	f	1	II	5	V	V	V	V	V
7	g	1	II	3	V	V	VI	-	-
8	h	1	II	1	VI	-	-	-	-
9	i	1	II	4	VI	VI	V	V	-
10	j	1	VII	1	VI	-	-	-	-
11	k	0	V	0	-	-	-	-	-
12	l	0	VI	0	-	-	-	-	-

The out-arities of these generators vary between 0 and 5, the objects with $\omega_{out} = \ell = 0$ consisting of the two generators k and ℓ.

The levels of abstraction vary between $\ell = 0$, for the two objects, to $\ell = 3$ for a and b. The two partitions $\{G_\mu^\omega\}$ and $\{G_\nu^\ell\}$ are given in Tables 4.3 and 4.4.

The bond values are from the sets $\mathscr{B}_\nu$ as described in Section 3.4.6 and tabulated in Table 4.2. One need not specify the in-bond values of the generators of maximum level of abstraction since no out-bond can connect to them. This is why the cells for β_{in} are empty for the two first rows of Table 4.1.

When the out-arity exceeds one we need generator coordinates to keep bonds apart, and this has been done by numbers 1,2,... marking the out-bonds. In-bonds are treated as identical, for any given generator, and therefore need no coordinates.

TABLE 4.2

SET FAMILY $\mathscr{B}$ FOR $\mathscr{T}_1$	
$\mathscr{B}_0$	{V, VI}
$\mathscr{B}_1$	{II, IV, VII}
$\mathscr{B}_2$	{I, III}

TABLE 4.3

ν	G_ν^ℓ
0	{k,ℓ}
1	{f,g,h,i,j}
2	{c,d,e}
3	{a,b}

TABLE 4.4

μ	G_μ^ω
0	{k,ℓ}
1	{b,c,d,e,h,j}
2	{a}
3	{g}
4	{i}
5	{f}

This is of course a quite small generator space, Actually, if we calculate the generator classes, putting the generator index equal to a constant, we see that each generator constitutes its own generator class, with the exception that d

and e belong to the same class $\alpha(d) = \alpha(e)$.

For this reason we have a poor group of similarities, only allowing the permutation of d and e. This is not a typical situation, but occurred since we chose a very simple example.

4.2.2. We can now combine the generators a,b,c,... following the rules $\mathscr{R}$ of combinatory regularity. We get, for example the regular configuration (a) in Figure 4.1 consisting of the generators k, occurring twice, g,ℓ, and j. It contains the objects, of type k and ℓ, and is of abstraction level one due to the occurrence of the generators g and j. To prove that (a) satisfies $\mathscr{R} = \langle \rho, \Sigma \rangle$ we first note from Table 4.1 that $\omega_{out}(k) = \omega_{out}(\ell) = 0$, so that k and ℓ have no out-bonds. Also that g has three out-bonds, all indicated in the diagram, with coordinates shown and j has one out-bond. Since all these four out-bonds are closed in the diagram the connection type Σ is satisfied. Now look at the four closed bonds in the diagram, for example the one from g to the upper k. From 4.1 we find that the first out-bond of g has bond value V, and that the in-bond value of k is V, equality holds and ρ is satisfied for this bond. In the same way we check the other bonds and conclude that $\mathscr{R}$ holds: (a) is a regular configuration. It is not a simple g-configuration since j can be dropped without destroying the regularity. However, with j deleted, the resulting subconfiguration is g-simple.

Another example is given in (b) with n(c) = 7. It consists of the generators k, occurring twice, e, twice, and g, h, and ℓ. It is of abstraction level two, due to the

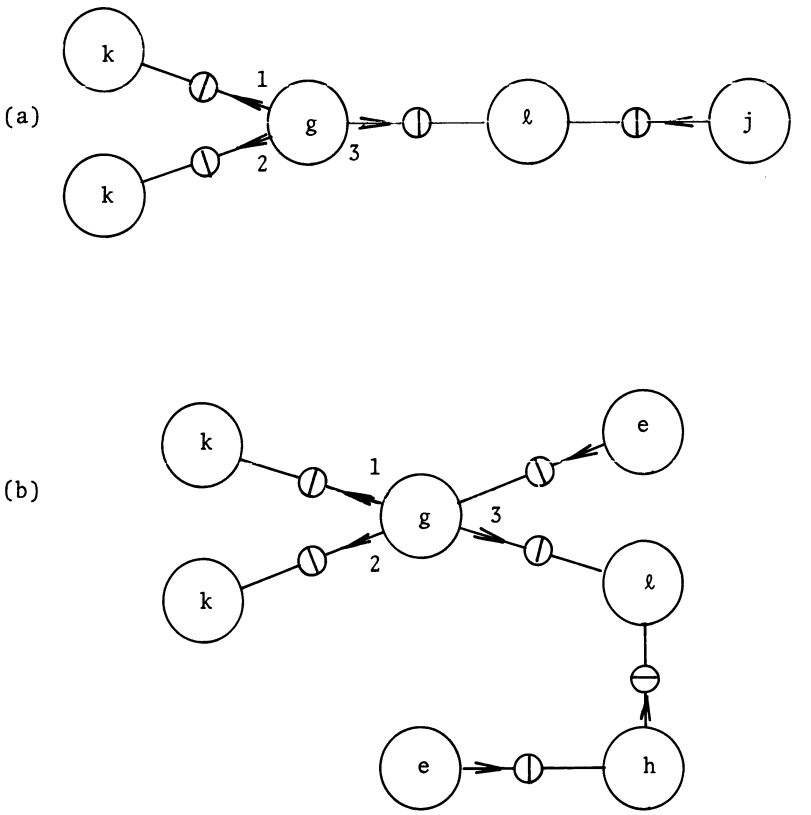


Figure 4.1

CONFIGURATION DIAGRAMS OVER G_1

occurrences of e. If we delete the bottom e and h we get an e-simple subconfiguration.

4.3.1. Let us now look at a more interesting example with the 34 generators listed in Table 4.5 together with their levels, out-arities, and bond values.

In addition the table contains identifiers, but now in the form of English word(s), intended to give the reader a concrete idea of the purpose of this regular structure. The

TABLE 4.5

GENERATORS FOR $\mathcal{I}_2$

number	identifier	level	β_{in}	ω_{out}	β_{out1}	β_{out2}	β_{out3}	β_{out4}	β_{out5}
1	individual 1	3	-	2	H	H	-	-	-
2	individual 2	3	-	2	H	H	-	-	-
3	left	3	-	1	H	-	-	-	-
4	right	3	-	1	H	-	-	-	-
5	horizontal	3	-	1	H	-	-	-	-
6	vertical	3	-	1	H	-	-	-	-
7	arm	2	H	1	G	-	-	-	-
8	idle	3	F'	1	G	-	-	-	-
9	strongly	2	F'	1	C	-	-	-	-
10	weakly	2	F'	1	C	-	-	-	-
11	clockwise	2	F'	1	D	-	-	-	-
12	counterclockwise	2	F'	1	D	-	-	-	-
13	left'	2	F'	1	G	-	-	-	-
14	right'	2	F'	1	G	-	-	-	-
15	horizontal'	2	F'	1	G	-	-	-	-
16	vertical'	2	F'	1	G	-	-	-	-
17	down	2	F'	1	E	-	-	-	-
18	up	2	F'	1	E	-	-	-	-
19	hand	1	G'	5	B	B	B	B	B
20	grasp	1	C	3	B	B	A	-	-
21	rotate	1	D	1	A	-	-	-	-
22	press	1	E	4	A	A	B	B	-
23	brass	1	F	1	A	-	-	-	-
24	steel	1	F	1	A	-	-	-	-
25	little	1	F	1	A	-	-	-	-
26	big	1	F	1	A	-	-	-	-
27	thumb	0	B	0	-	-	-	-	-
28	index finger	0	B	0	-	-	-	-	-
29	middle finger	0	B	0	-	-	-	-	-
30	ring finger	0	B	0	-	-	-	-	-
31	little finger	0	B	0	-	-	-	-	-
32	bolt	0	A	0	-	-	-	-	-
33	nut	0	A	0	-	-	-	-	-
34	cylinder	0	A	0	-	-	-	-	-

idea is to formalize hand motions of two individuals working with a few things made of metal. This should be compared to the discussion in Section 3.7 of Volume I on motion studies, and to Case 3.6.4 (anatomical patterns), ibid.

A few remarks will be in order. The in-bond values F and F' never occur among out-bond values. This implies that the generators 8-18 and 23-26 never accept out-bonds from other generators.

We have two generators *right* and *right'* that seem to play the same role and analogously for *left* and *left'*. This is not so, however. The generator g = *right* connects to g = *arm*, but not to *hand*. The generator *right'*, on the other hand, connects to *hand*, but not to *arm*. We have thought of an arm as being made up of various parts, one of them being a hand. It is then not appealing to allow the same unary relation (property) to be applicable to both, on various levels of abstraction.

The vagueness of every day language tends to conceal such subtle semantic distinctions, but one of the advantages of the abstract approach is that it forces precision upon us.

This is of more general significance than may be immediately obvious. However, if we decided, against the reasoning just given, that a generator, for example *right*, should be allowed to connect to generators of different levels of abstraction we would have to modify the assumptions in 3.4.6. This can certainly be done, and with little effort, but seems unnatural.

The generator *grasp*, of out-arity three, should be read as "grasp something$_3$ between finger$_1$ and finger$_2$". The markers are the bond coordinates. The generator *press*, of out-arity

four, should be read "press something$_1$ against something$_2$ using finger$_3$ and finger$_4$. The remaining generators need no explanation.

4.3.2. The set families $\mathscr{B}_\nu$ are given in Table 4.6, and the partitions according to level and out-arity in Tables 4.7-4.8 respectively.

The index classes are easily calculated; see the discussion in Section 3.5. There are 15 of them given in Table 4.9.

TABLE 4.6

	SET FAMILY $\mathscr{B}_\nu$ FOR $\mathscr{T}_2$
$\mathscr{B}_0$	{A,B}
$\mathscr{B}_1$	{C,D,E,F,G}
$\mathscr{B}_2$	{H,F'}

TABLE 4.7

ν	G_ν^ℓ
0	27,28,29,30,31,32,33,34
1	19,20,21,22,23,24,25,26
2	7,8,9,10,11,12,13,14,15,16,17,18
3	1,2,3,4,5,6

The reader may be interested in recognizing the common characteristics, in every-day language, of generators with the same index α. It is also of interest to compare the index classes with the map μ in Table 4.9 and the relation $\beta \to \beta'$ in Table 4.11 given later in the text.

TABLE 4.8

μ	G_μ^ω
0	27,28,29,30,31,32,33,34
1	3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,21,23,24,25,26
2	1,2
3	20
4	22
5	19

TABLE 4.9

α	G^α	$\#(G^\alpha)$
1	{1,2}	2
2	{3,4,5,6}	4
3	{7}	1
4	{8}	1
5	{13,14,15,16}	4
6	{9,10}	2
7	{11,12}	2
8	{17,18}	2
9	{19}	1
10	{20}	1
11	{21}	1
12	{22}	1
13	{23,24,25,26}	4
14	{27,28,29,30,31}	5
15	{32,33,34}	3
		34

The similarities are now many more than in the first example. The group S of similarities is the direct product of full symmetric groups of order 2,4,1,1,4,...; see the

last table. Hence

$$\#(S) = 2!4!... \cong 1.6 \cdot 10^8. \qquad (4.1)$$

4.3.3. Combining the generators *thumb, index finger, grasp, bolt, brass* we get the regular configuration in Figure 4.2(a). Another one is shown in (b). A more complicated case is given in (c) with $n(c) = 13$. The sub-configuration c' inside the dotted contour is regular, and can be thought of as a macro-generator, see Volume I, p. 32. The whole configuration is of abstraction level three.

Two macro-generators c'' and c''' appear in (d), where two individuals are at work together. This configuration is also of abstraction level 3.

The configuration c_1 in the figure, in (e), is simply related to the one c_2 in (a): they are similar, $c_1 = sc_2$, $s \in S$.

Regular configurations as above, and the resulting images in $\mathscr{I}_2$, describe hand motions of one or two individuals. It would be misleading, however, to say that such images *mean* certain motions in the physical world. To do this we would need *another* formalization of the physical worlds on the level of the natural sciences. In our way of thinking semantics means a correspondence between two regular structures.

As pointed out in Section 2 we do not necessarily assume that the perception of the world of our observer is logically consistent. As a matter of fact the term "logically consistent" requires that second regular structure just mentioned, which may be absent. Without introducing it we should not worry too much if we encounter images in $\mathscr{I}_2$ with individuals with five thumbs, or two individuals sharing an arm.

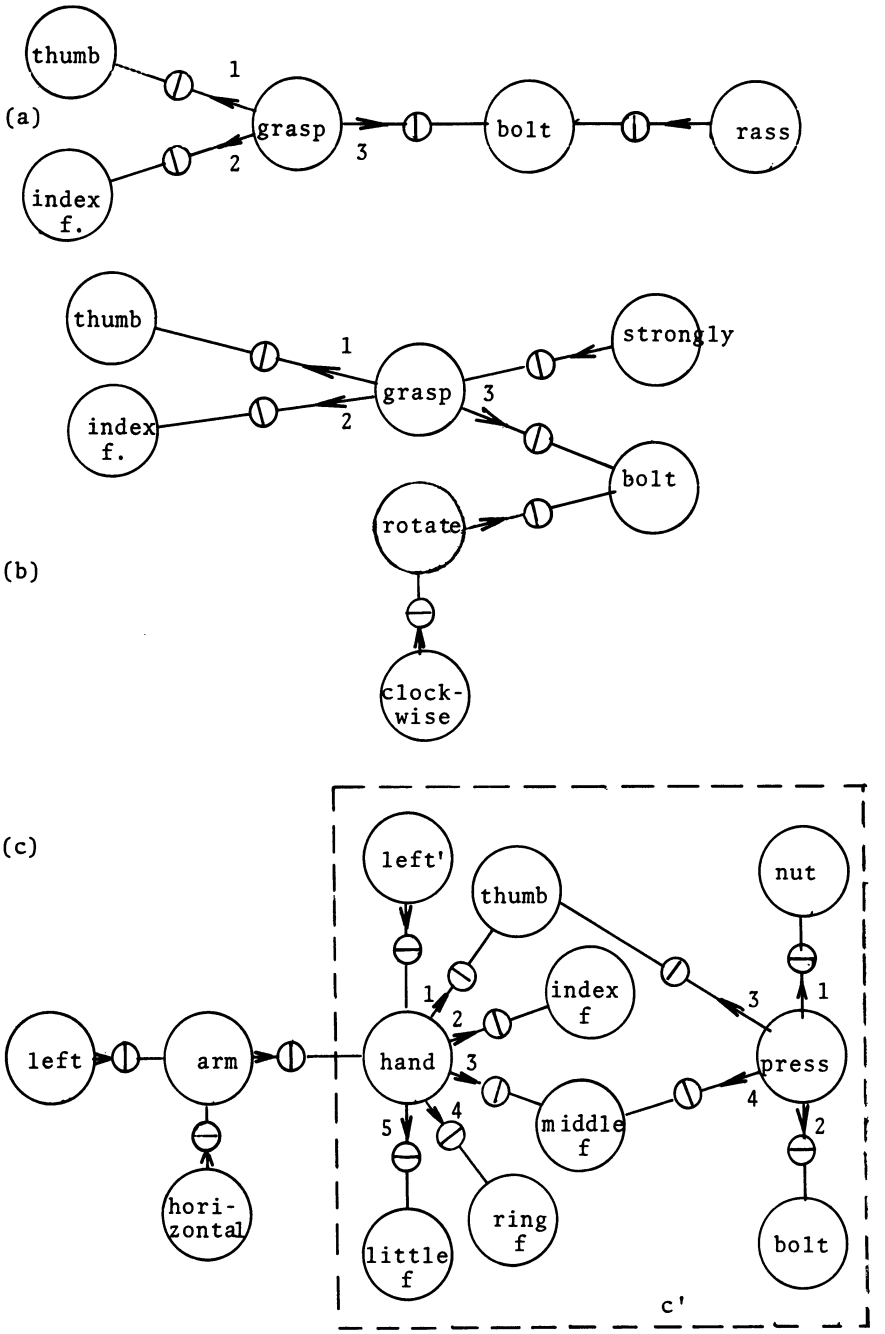


Figure 4.2

CONFIGURATION DIAGRAMS OVER G_2

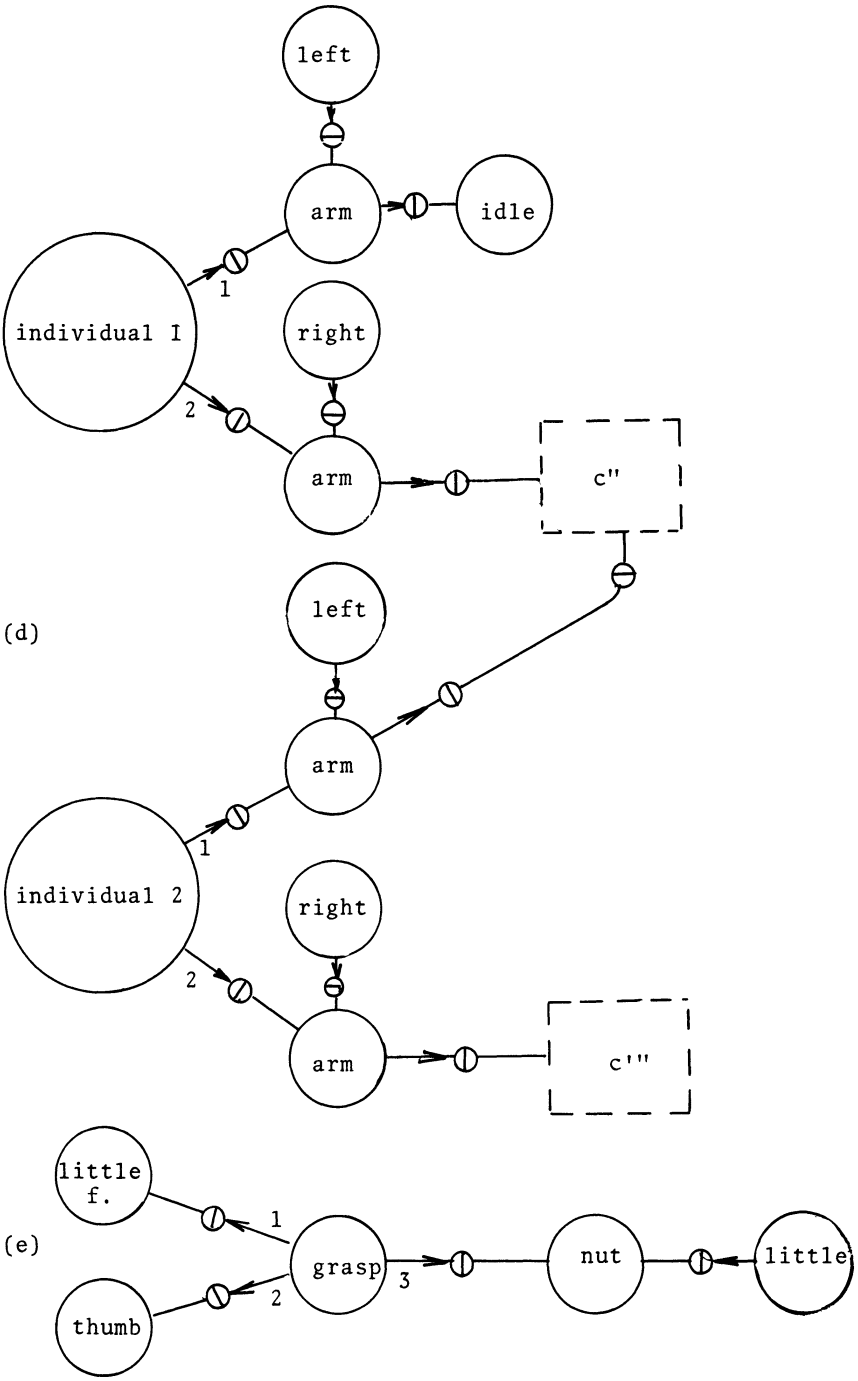


Figure 4.2 (cont.)

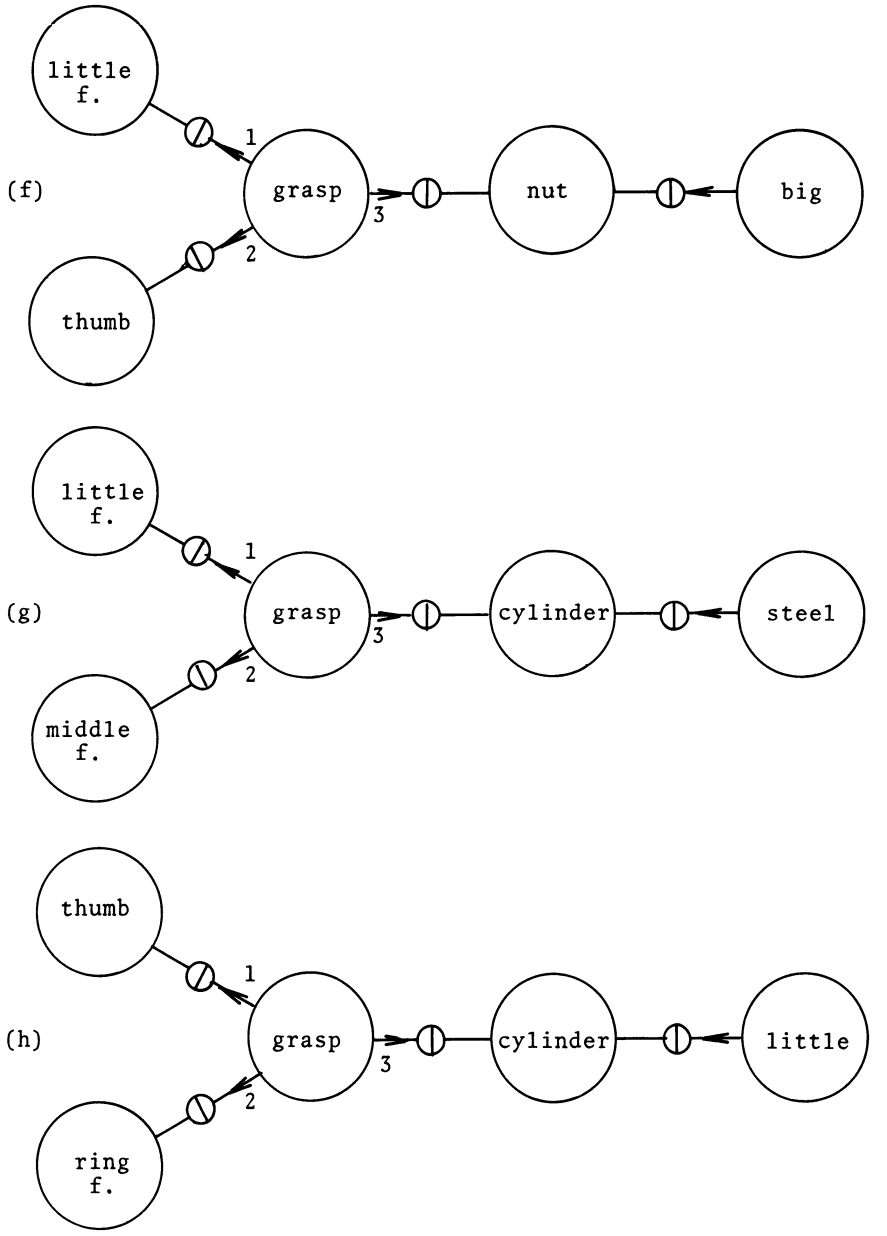


Figure 4.2 (cont.)

If we want to remove such images from the algebra it can be done by labelling in-bonds by markers; at the present stage this does not seem necessary.

The last configurations (f), (g), (h) are similar and belong to the same pattern as the one in (a). This can be checked using Table 4.10 specifying the similarity group S.

4.4. To establish a correspondence between our two configuration spaces (as well as the related image algebras) we consider the following generator map $\mu: G_2 \to G_1$ together with the table of the associated bond value map $\beta \to \beta'$; see Tables 4.10-4.11.

TABLE 4.10

g	μg	g	μg	g	μg
1	a	13	e	25	j
2	a	14	e	26	j
3	b	15	e	27	k
4	b	16	e	28	k
5	b	17	e	29	k
6	b	18	e	30	k
7	c	19	f	31	k
8	d	20	g	32	ℓ
9	e	21	h	33	ℓ
10	e	22	i	34	ℓ
11	e	23	j		
12	e	24	j		

TABLE 4.11

β	β'	β	β'
A	VI	E	II
B	V	F	IV
C	II	F'	III
D	II	G	II
		H	I

Referring to Remark 2 in 3.7.9 we can verify that μ generates a homomorphism h: $\mathscr{C}_2 \to \mathscr{C}_1$. We just have to check that, (a), μ preserves bond-structure, $B^S(\mu g) = B^S(g)$, using Table 4.9 together with Tables 4.5 and 4.1, and that, (b), the bond values behave as required in the quoted Remark 2.

Applying this homomorphism for example to (a) in Figure 4.2, we get the $\mathscr{C}_1$-configuration (a) in Figure 4.1. In the same way the $\mathscr{C}_2$-configuration (b) in Figure 4.2 is carried over into (b) in Figure 4.1.

As typical for homomorphisms, h loses information: a $\mathscr{C}_1$-configuration (or one in $\mathscr{C}_1$) is less informative, although topologically the same, compared to a $\mathscr{C}_2$-configuration (or one in $\mathscr{C}_2$).

9.5. The choice of language type for the study

5.1.1. In accordance with the discussion in Section 2, and with the stated reservations, we shall choose the type of language to be used by the speaker/listener as *finite state*. We can be quite brief when discussing these languages, they are so well known.

5.1.2. The grammar $\mathscr{G}$ will be based on a vocabulary $V = V_T \cup V_N$. Here V_T is the *terminal vocabulary* consisting

of the *words* to be used. We shall sometimes use Greek or Roman letters to denote the elements of V_T, and occasionally common words in English. In the latter case we have to watch out so that we do not forget that *they should be treated abstractly*, not representing a subset of real English.

The *non-terminal vocabulary* V_N consists of *syntactic variables, or states*. They will be denoted by numbers $1,2,3,\ldots F$, where F indicates the final state. We shall use the convention that 1 is the start state.

5.1.3. The *productions* in $\mathscr{G}$ can always be expressed in canonical form as

$$i \overset{x}{\rightarrow} j, \quad x \in V_T, \quad i,j \in V_N. \tag{5.1}$$

We can read (5.1) as "the state i goes into j while writing the terminal symbol x". Sometimes it will be convenient to let x in (5.1) indicate a finite string instead, $x \in V_T^*$, but this will not affect the generative power of the grammar.

5.1.4. Just as in Volume II, Chapter 8, we shall assume that the grammar has been reduced to deterministic form so that any productions in $\mathscr{G}$ of the form

$$\begin{cases} i \overset{x}{\rightarrow} j \\ i \overset{x}{\rightarrow} k \end{cases} \tag{5.2}$$

must coincide, $j = k$. This makes parsing of sentences unambiguous, so that if $x_1x_2x_3\ldots x_n$ is grammatical it has a unique parsing into

$$\overset{1}{x_1}\overset{i_1}{x_2}\overset{i_2}{x_3}\ldots\overset{F}{x_n} \tag{5.3}$$

In (5.3) we have parsed the sentence into successive produc-

tions $1 \overset{}{\underset{x_1}{\to}} i_1$, $i_1 \overset{}{\underset{x_2}{\to}} i_2, \ldots i_{n-1} \overset{}{\underset{x_n}{\to}} F$.

5.1.5. The set $L \subseteq V_T^*$ of finite strings produced like this constitutes the *language* generated by the language $L = L(\mathscr{G})$.

With the same procedure for producing strings, but not requiring that the state $i_o = 1$ or $i_n = F$, we get *grammatical phrases*

$$i_o \, x_1 \, i_1 \, x_2 \, i_2 \, x_3 \ldots x_n \, i_n \, . \tag{5.4}$$

They need not belong to the language, but can be of linguistic significance anyway.

5.2. Equivalently the language can be represented by a *finite automaton* that we shall often present in diagrammatic form. To clarify the notation consider the finite automaton given by the simple wiring diagram in Figure 5.1.

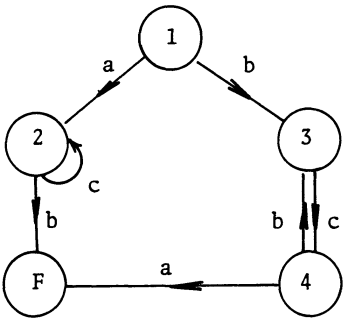


Figure 5.1

The corresponding grammar has

$$\begin{cases} V_T = \{a,b,c\} \\ V_N = \{1,2,3,4,F\} \end{cases} \tag{5.5}$$

and the productions in the table

TABLE 5.1

$1 \overset{a}{\rightarrow} 2$	$3 \overset{c}{\rightarrow} 4$
$1 \overset{b}{\rightarrow} 3$	$4 \overset{b}{\rightarrow} 3$
$2 \overset{c}{\rightarrow} 2$	$4 \overset{a}{\rightarrow} F$
$2 \overset{b}{\rightarrow} F$	

It is obviously deterministic. It generates for example sentences parsed as

$$\begin{cases} 1_a 2_c 2_c 2_b F \\ 1_b 3_c 4_b 3_c 4_a F \\ 1_b 3_c 4_a F \end{cases} \tag{5.6}$$

and phrases like

$$\begin{matrix} 2_c 2_c 2 \\ 3_c 4_b 3_c 4 \end{matrix} \tag{5.7}$$

5.3. Another equivalent way of representing finite state languages is via *regular expressions* from formal logic. Such expressions are built from concatenation, finite repetition (indicated by a star), union, and parentheses to indicate order of execution.

The language generated by the wiring diagram in Figure 5.1, for example, can then be written as

$$L = (ac^*b) \cup (bc(bc)^*a). \tag{5.8}$$

It is clear $\#(L) = +\infty$ if and only if the regular expression contains at least one star. This is the only case of interest for us and will be assumed throughout this paper.

5.4.1. For the following it is of paramount importance that *finite state languages*, as well as many other formal languages, *can be viewed as combinatory regular structures;* see Volume I, Sections 2.4 and 3.2.

The generators will then be represented by the productions of $\mathscr{G}$ (not by words!) They have $\omega_{in} = \omega_{out} = 1$, with the in-bond value given by the state i to be rewritten in (5.1) and the out-bond value as j, the resulting state. Further ρ = EQUAL and Σ = LINEAR.

The identification rule R to be used then identifies two regular configurations (grammatical phrases) if they consist of the same string of terminal symbols and have the same external in-bond value and the same external out-bond value. Remember that the finite automaton was here assumed to be deterministic, then each image consists of a single configuration.

5.4.2. Strictly speaking, this image algebra represents not just L, but all grammatical phrases in L. When we want to limit ourselves to just L, we need two more generators, one g' with $\omega_{in}(g') = 0$, $\omega_{out}(g') = 1$ with out-bond value 1, and another one, g^F, with $\omega_{in}(g^F) = 1$, $\omega_{out}(g^F) = 0$ with the in-bond value F. We shall then let $\mathscr{T}_2$ be the image algebra consisting of all images in $\mathscr{T}$ with the connection type LINEAR with the additional constraint that the out-arity be zero. $\mathscr{T}_2$ will reappear in the next section as the secondary image algebra in semantic relations.

9.6. Semantic maps

6.1.1. We shall now try to formalize in algebraic form the ideas on semantics from Section 2. To begin with we shall do this in a fairly general setting, attempting to bring out clearly the major problems that confront us in our task. Gradually we shall specialize by bringing in restrictions on the semantic maps, and in Section 7 we will examine the detailed structure of some semantic schemes.

6.1.2. In our view *semantics is relative: it relates two or more regular structures to each other.* Consider therefore two image algebras

$$\begin{cases} \mathscr{I}_1 = \langle \mathscr{L}_1, R_1 \rangle \\ \mathscr{I}_2 = \langle \mathscr{L}_2, R_2 \rangle. \end{cases} \tag{6.1}$$

We want to "explain" $\mathscr{I}_2$ in terms of $\mathscr{I}_1$ by relating images from $\mathscr{I}_2$ to some in $\mathscr{I}_1$. To distinguish between them, let us speak of $\mathscr{I}_1$ as the *primary image algebra* and of $\mathscr{I}_2$ as the *secondary image algebra.*

The *semantic map*, $\underline{\text{sem}}$: $\mathscr{I}_2 \to \mathscr{I}_1$, defined on some subset $\mathscr{I}_2' \subseteq \mathscr{I}_2$, shall be uniquely defined. Otherwise our "explanation" would be ambiguous. This is something we have decided to avoid; see Section 2.4. We shall then say that $\underline{\text{sem}}$ is *adequate for* $\mathscr{I}_2$ *relative to* $\mathscr{I}_1$.

The inverse of $\underline{\text{sem}}$ need not be unique. A given primary image I can correspond to a set (with several elements)

$$\underline{\text{sem}}^{-1}(I) \subseteq \mathscr{I}_2. \tag{6.2}$$

Sometimes it is better to start the analysis of a semantic scheme via this inverse map $\underline{\text{sem}}^{-1}$: $\mathscr{I}_1 \to 2^{\mathscr{I}_2}$. When the

secondary image algebra is a language, $\underline{\text{sem}}^{-1}(I)$ consists of all grammatical utterances that "mean" I, and $\underline{\text{sem}}^{-1}$ expresses the *linguistic strategy* of the speaker; more about this in Section 7.

The following terminology will be used. If sem is defined on the whole of $\mathscr{T}_2$ it is said to be *entire*. If sem: $\mathscr{T}_2 \to \mathscr{T}_1$ is surjective *and* entire it is said to be *perfect*.

6.2.1. In this section we shall always let the primary image algebra be a relation image algebra, as discussed in Section 3. The secondary image algebra shall consist of a finite state language $L(\mathscr{G})$, viewed as a regular structure; see 5.4.

Since we have $\text{card}(\mathscr{T}_1) = \text{card}(\mathscr{T}_2)$, both being denumerably infinite, there is of course no problem with the existence of semantic maps adequate for $\mathscr{T}_2$ relative to $\mathscr{T}_1$. Indeed, there always exist bijective maps $\mathscr{T}_1 \leftrightarrow \mathscr{T}_2$, and infinitely many of them.

6.2.2. The trouble is that a semantic map, even though adequate, even bijective, is of little interest unless it has additional structure. If it is given just as a list of pairs (I_2,I_1), $I_2 \in \mathscr{T}_2$, $I_1 \in \mathscr{T}_1$, and with no more a prioristic knowledge, it could not possibly be learnt from finite experience, nor could it be remembered using a finite memory.

To supply this missing structure we shall exploit the combinatory regularity of the two image algebras.

6.3. Let us approach this topic from a trivial example. Say that G_1 consists entirely of objects. Then any image in $\mathscr{T}_1$ is just a set of generators each of which has outarity zero, so that they cannot be connected to each other.

To describe such primary images it suffices to introduce the finite state language with $V_T = G_1$, $V_N = \{1,F\}$ and all productions of the form $1 \overset{g}{\to} F$ or $1 \overset{g}{\to} 1$, $g \in G_1$. This language has the regular expression G_1^*.

If $I_2 = g_1,g_2,\ldots g_n$ with $n(g)$ occurrences of the word g, $g \in G_1$, then the semantic map

$$\text{sem}(I_2) = \text{image with } n(g) \text{ generators } g, \quad g \in G_1 \qquad (6.3)$$

is obviously adequate. The inverse $\text{sem}^{-1}(I_1)$ gives us all strings over G_1 of length n with $n(g)$ occurrences of g, in arbitrary order; it is not unique.

6.3.2. A reader is certainly entitled to object to this example being too simple-minded: real semantics is infinitely more complicated. And this is just why we picked the example. As soon as we allow connections in the primary image algebra, syntactic constraints will be forced upon us in order to make the semantics adequate.

Consider another example, still quite simple, with the generators in G_1 given in Table 6.1. An arbitrary image in $\mathscr{I}_1$ then consists of, say, r α-generators, to $m_{11},m_{12},\ldots,m_{1r}$ of which a γ-generator connects respectively, in addition to s β-generators, to $m_{21},m_{22},\ldots m_{2s}$ of which δ-generators are attached; see Figure 6.1 where $r = 3$, $m_{11} = 1$, $m_{12} = 0$, $m_{13} = 2$ and $s = 2$, $m_{21} = 3$, $m_{22} = 0$.

A language suitable to describe such images is easy to construct and we exhibit one in terms of the wiring diagram of its finite automaton in Figure 6.2.

This language will be supplied with a semantic map as follows. Given a grammatical sentence, for each time we pass

TABLE 6.1

number	identifier	level	β_{in}	ω_{out}	β_{out}
1	α	0	A	0	-
2	β	0	B	0	-
3	γ	1	-	1	A
4	δ	1	-	1	B

the branch $2 \overset{a}{\to} 3$ we add a generator α to the configuration diagram. For each time we pass the branch $3 \overset{g}{\to} 3$ we add and connect a generator γ to the last α introduced. Each time we pass the branch $4 \overset{b}{\to} 5$ we add a generator β to the configuration, and for each pass through $5 \overset{d}{\to} 5$ we add and connect a generator δ to the last β. For other branches in the figure we do not modify the configuration.

This defines $\mathrm{sem}(I_2)$, uniquely on $\mathscr{T}_2$; it is adequate for $\mathscr{T}_2$ relative to $\mathscr{T}_1$. It is also clear that sem is surjective. Given a primary image I_1 let us enumerate its generators of level 0, first the α's and then the β's. After any occurrence of an α we put the γ's attached to it; similarly for the β's and δ's. With this arrangement pass through the diagram in Figure 6.2 writing terminal symbols successively. If no α is present in I_1, we go to 4, writing a y; otherwise to 2 writing an x, and then to 3 writing an a. If the α has one or several γ's attached, loop through $3 \overset{g}{\to} 3$ the same number of times. If any more α is present go back to 2 and so on; else go to 4 and behave in the same way.

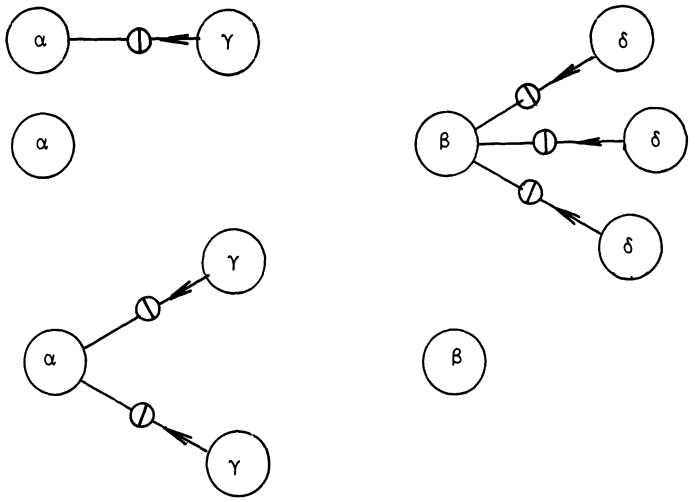


Figure 6.1

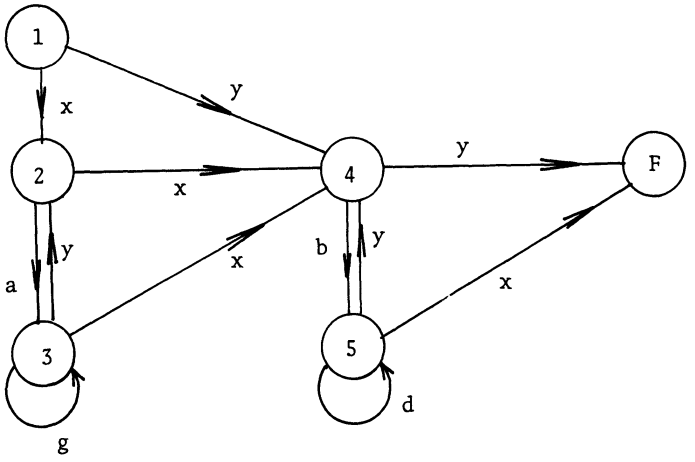


Figure 6.2

Here V_T = {a,b,d,g,x,y}, V_N = {1,2,3,4,5,F} and the productions are the rewriting rules $1 \overset{x}{\rightarrow} 2$, and so on, along the branches.

This will produce a grammatical sentence I_2, $I_2 \in \text{sem}^{-1}(I_1)$. For Figure 6.1, for example, we get the sentence

$$I_2 = \text{xagyayaggxbdddybx}. \tag{6.4}$$

This semantic map is perfect. The example is somewhat misleading, however, as will be seen later.

6.4.1. In the sentence (6.4) the words a,b,g,d indicate, in the way we have described, the occurrence of the "related" generators $\alpha,\beta,\gamma,\delta$. The other words x,y play a different role, they indicate what connections are established between the relations (G_1-generators).

One could let V_T consist of two disjoint sets V_{name} and V_{conn}. When we do this, each $g \in G_1$ shall correspond to a set $\underline{\text{name}}(g) \subseteq V_{\text{name}}$. The inverse $\underline{\text{name}}^{-1}$ tells us which generator a name $\in V_{\text{name}}$ represents. The words in V_{conn}, the *connectives* (used in a different sense from ordinary syntax), *are needed for carrying topological information.*

6.4.2. It should be unnecessary to warn the reader that this is not supposed to model natural language, where no such clearcut distinction between names and connectives can be made. Our view is entirely abstract, and speculative rather than empirical.

6.4.3. The last example brings out what is the essential difficulty in establishing semantic maps. Finite state language, considered as a regular structure, has connection type Σ_2 = LINEAR; see 5.4.1. Our relation image algebra on the other has a much more flexible connection type Σ_1.

Although we shall stay with finite state languages we cannot avoid reminding the reader that with *context free*

languages we get a more powerful topology, namely Σ = TREE; see Volume I, Section 2.6. Still more powerful is the connection type for *context-sensitive* languages which allows cycles, just as in our primary image algebra. This should be looked into more carefully in our future work.

6.5.1. The last example contains a clue for the understanding of semantic maps more generally. To make this clear let us return to the wiring diagram in Figure 6.2. For a given grammatical sentence I_2 we start with the empty configuration at state 1. If the first word in I_2 is x we go to 2, keeping the configuration empty. If the next word in I_2 is a, then we go to state 3 and add α to the empty configuration. If the next word in I_2 is g then we connect the generator γ to the generator in our monatomic configuration. We now have a biatomic configuration, this is processed, and we continue until we have reached and used the last word in I_2. Then we have a configuration c from $\mathscr{C}_1$: the corresponding image $R_1(c) = \underline{sem}(I_2)$.

This is a sequential process, mapping configurations in $\overline{\mathscr{C}}_1$ into others in the same space. Which configuration map will be applied during each step of the process depends upon what branch we are passing through in the wiring diagram.

6.5.2. This leads us to a concept that is fundamental for the following analysis.

Definition 6.1. *By a semantic* (finite-state) *processor from* $\mathscr{C}_2$ *to* $\mathscr{C}_1$ *we shall mean a collection of sets* $C_i \subseteq \overline{\mathscr{C}}_1$, *with* $C_1 = \phi$, $C_F = \mathscr{C}_1$, *with connectors* $\sigma_{ij}(x)\colon C_i \to C_j$; $x \in V_T$, (*where* σ_{ij} *stands for a connector that may or may not contain new generators*)

a) *We start at state* 1 *with* c = ϕ

b) *A sentence* $x_1x_2\ldots x_n \in L(\mathscr{G})$ *is processed left-to-right*

c) *At any transition back to state* 1 c *is made equal to* ϕ *again.*

Remark. The statement in c) means that we start to build a new subconfiguration, starting from the empty one. The new one will not be connected to the one(s) already constructed. Rule c) is less essential than a) and b) and can be left out.

Although we shall study only finite state languages in this paper, the definition has been formulated in such a way that it should be possible to adapt it for more powerful languages.

6.5.3. To gain some intuitive understanding of the role of this definition, let us return to the example in Section 6.3. Introduce the subsets of $\mathscr{C}_1$

$$\begin{cases} C_1 = \phi \\ C_2 = \text{configurations with r } \alpha\text{'s, r} \geq 0\text{, for each of which may be attached a number of } \gamma\text{'s} \\ C_3 = \text{same as } C_2 \text{ except that r} \geq 1 \\ C_4 = C_2 \\ C_5 = \text{same as } C_2 \text{ but followed by at least one } \beta\text{, all the } \beta\text{'s may have } \delta\text{'s attached or not} \\ C_F = \mathscr{C}_1. \end{cases} \tag{6.5}$$

In this example all the C_i-classes consist of regular $\mathscr{C}_1$-configurations, but this need not always be true. More about this later.

The associated configuration maps given by connectors

$\sigma_{ij}(x)$ are defined if $i \overset{x}{\to} j$ is a branch in the diagram

$$\begin{cases} \sigma_{12}(x) = \sigma_{13}(y) = \sigma_{24}(x) = \sigma_{22}(y) = \sigma_{34}(x) = \sigma_{44}(y) \\ \qquad = \sigma_{54}(y) = \sigma_{55}(F) = \text{identity operation} \\ \sigma_{23}(a) = \text{add unconnected } \alpha \text{ to configuration} \\ \sigma_{33}(g) = \text{connect new } \gamma \text{ to last } \alpha \\ \sigma_{45}(b) = \text{add unconnected } \beta \text{ to configuration} \\ \sigma_{55}(d) = \text{connect new } \delta \text{ to last } \beta. \end{cases} \tag{6.6}$$

Remark 1. Since we interpret branching back to state 1 as meaning "begin a new (unconnected) component of the configuration to be calculated", we could have let, for example, the branch $3 \overset{x}{\to} 4$ to 1 instead. Remember that to describe unions of unconnected sub-configurations we need no syntactic information in addition to what is already contained in the sentence to describe the sub-configurations.

Remark 2. In the successive evolution of the c's we may have to refer to generators and bonds, which will be done in terms of the configuration coordinates.

Remark 3. The processor used is related to the concept of tree automata.

6.6.1. Given a semantic processor $\mathscr{G}_2 \to \mathscr{G}_1$ we can extend the configuration maps $\sigma_{ij}(x)$ to be defined for phrases u (in $L(\mathscr{G})$) by putting $\sigma_{ij}(u)$ = undefined if u is not grammatical, and if u is the grammatical phrase (5.4) with $i_o = i, i_n = j$, put

$$\sigma_{ij}(u) = \sigma_{i_{n-1}i_n}(x_n) \cdots \sigma_{i_1 i_o}(x_2)\sigma_{i_o i_1}(x_1). \tag{6.7}$$

Due to associativity (6.7) is well-defined, and since $\mathscr{G}$ is deterministic the string $i_o, i_1, i_2, \ldots i_n$ is unique and hence

also (6.7). To the empty string $u = \phi$ we associate $\sigma_{ij}(u) =$ identity.

6.6.2. With the extended semantic map, the configuration map σ_{1F} represents our semantic map for configurations. We get for the image algebras after R_1-identification has been carried out in $\mathscr{T}_1$

$$\mathscr{T}_2 \to \mathscr{T}_1\colon \underline{\text{sem}}(I_2) = R_1[\sigma_{1F}(I_1)]. \tag{6.8}$$

6.6.3. We have obtained the semantic map by *sequentially unwrapping the meaning of the given sentence.* This should be compared with the way Wegner (1968) views executing a program as the successive transformation of information structures. In the present case the information structures are configurations from $\mathscr{C}_1$. At each step old bonds may be closed and new generators be added. The out-bonds of the new generators may be left open or be closed immediately. In the example the latter was the case.

The semantic processor still involves operations of too general a nature. In the next section we shall narrow down the choice further.

6.7. Before doing this we mention the following simple observation that serves to bring out more clearly the algebraic structure of the problem of mathematical semantics.

Theorem 1. *The extended semantic processor forms a category.*

Proof: Introduce the objects (in the terminology belonging to categories) C_i and the classes, possibly empty, of morphisms $C_i \to C_j$

$$K_{ij} = \{\sigma_{ij}(u)\ u \in V_T^*\}. \tag{6.9}$$

It is clear that K_{ii} contains the identity map $C_i \to C_i\colon \text{id}_{C_i}$.

The way we have extended the original semantic map in Definition 6.1 to V_T^* it follows directly that

$$\sigma_{ik}(u)\circ\sigma_{k\ell}(v) = \sigma_{i\ell}(uv) \in K_{i\ell} \qquad (6.10)$$

where uv stands for the concatenation of the strings u and v. Hence the semantic processor forms a category. Q.E.D.

6.7.1. Consider a configuration $c \in C_i$ with content(c) = $(g_1, g_2, \ldots g_n)$, subscripts are the coordinates of the generators, and bonds $\beta_{k\ell}$, $k = 1,2,\ldots n$, $\ell = 1,2,\ldots \omega_{out}(g_k)+1$. The in-bond of any g_k can be represented by a single value of ℓ, since the values and structural parameters of in-bonds (for one and the same generator) have been assumed to be the same.

Let u be an arbitrary grammatical phrase starting at the state i ending at j, and with the string of arbitrary finite length $x_1x_2\ldots \in V_T^*$. When we apply the corresponding connector $\sigma_{ij}(x_1x_2\ldots)$ to c some of c's bonds will be connected and the rest will not. Denote by $T_i(c)$ the table of the bonds belonging to c that can be connected for some grammatical phrase u starting at i.

Each entry of $T_i(c)$ will consist of three parts. One is the bond coordinate, another consists of bond-structure parameters, and the third is the bond value. During the sequential process we need only keep in memory content(c) and $T_i(c)$.

6.7.2. The memory requirement will therefore depend upon how large the tables content(c) and $T_i(c)$ are. The behavior of #[content(c)] is easy to find.

Lemma 1. *We have for* $c' = \sigma_{ij}({}^iu^j)c$

$$\text{content}(c') = \text{content}(c) \cup \text{content}[\sigma_{ij}({}^{i}u^{j})]. \qquad (6.11)$$

The proof is immediate, since connectors can only add, not delete generators.

The relation (6.11) implies that

$$\#(c') = \#(c) + \#[\sigma_{ij}({}^{i}u^{j})]. \qquad (6.12)$$

The behavior of the size of $T_i(c)$ depends on the particular semantic map and can differ drastically from case to case as will be seen in the next section.

9.7. Special semantic maps

7.1.1. For a given primary image algebra it is easy to construct a scheme that maps any regular configuration into a finite string over a finite vocabulary, in such a way that this string uniquely determines the configuration.

We shall illustrate how this is done via the image algebra in 6.3.2. Choose V_T as consisting of the symbols $\alpha,\beta,\gamma,\delta,I,II$. In other words, we use the elements in G_1 together with two demarcation symbols called I and II.

If $\text{content}(c) = (g_1,g_2,\ldots g_n)$ we start the string by $g_1g_2\ldots g_nII$. If the first out-bond of g_1 goes to g_i we concatenate the string $g_1g_2\ldots g_iI$. If the next out-bond goes to g_j we concatenate $g_1g_2\ldots g_jI$, and so on. After the last out-bond of g_1 we use the symbol II again, then continue with the out-bonds in g_2, and so on, until the entire configuration has been exhausted. When no out-bonds exist no symbol is used between occurrences of II. We do not use I at the end of the bonds of a generator.

The configuration in Figure 6.1, for example, will be mapped into the string

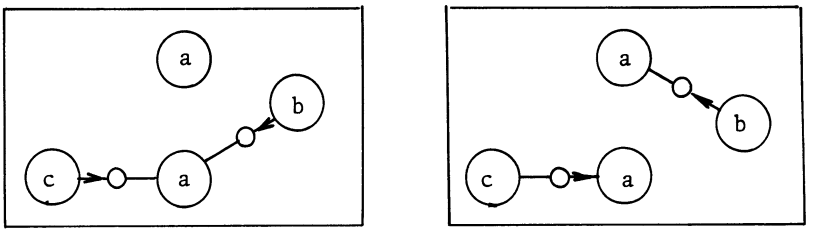
$$\alpha\gamma\alpha\alpha\gamma\gamma\beta\delta\delta\delta\beta\mathrm{IIII}\alpha\mathrm{IIIIII}\alpha\gamma\alpha\alpha\mathrm{II}\alpha\gamma\alpha\alpha\mathrm{IIII}\alpha\gamma\alpha\alpha\gamma\gamma\beta\mathrm{II}\ \mathrm{II}\alpha\gamma\alpha\alpha\gamma\gamma\beta\mathrm{II}\alpha\gamma\alpha\alpha\gamma\gamma\beta\mathrm{IIII}. \tag{7.1}$$

The decoding is easy. The substring before the first occurrence of II gives us content(c). The substrings between successive occurrences of II give us the out-bond connections of each generator. Recall that the out-arities are known from G_1, so the references will be unambiguous.

7.1.2. This will give us very long strings, even for simple configurations, such as in Figure 6.1.

We have not specified the syntax of the language. The language is certainly not the entire V_T^*, since most of the elements of this set are not coded representations of elements in $\mathscr{L}_1$. Instead the combinatory regularity $\mathscr{R}_1$ induces syntactic constraints for the coded strings.

We mention parenthetically the reason why we had to refer to generators by strings $g_1g_2\ldots g_i$, rather than just by g_i. If the configuration to be talked about contains two identical generators equal to g, say, the latter way of referring to them would be ambiguous. In Figure 7.1 the image in (a)



(a) (b)

Figure 7.1

clearly differs from the one in (b), although both are built on the same generator and with connection $c \to a, b \to a$. To specify that b be bonded to a is ambiguous since there are two a's.

This is not always the case, see Figure 7.2. Here it does not matter to which a-generator we connect the out-bond of b, since R_1 will identify the two resulting configurations.

If we could exclude situations like the one in Figure 7.1, we would make our task to construct adequate semantics easier. But that would be to avoid *a difficulty that seems to be intrinsic to the whole topic*, so that we have to face up to the problem in some way.

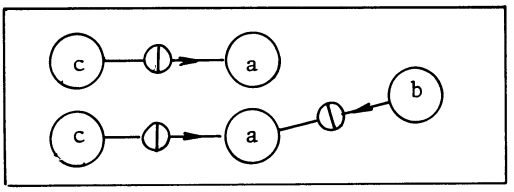


Figure 7.2

7.2.1. Rather than pursuing ad hoc schemes as the one illustrated by (7.1), it is more attractive to start from the other end, with a given semantic map, consider its memory requirement and relate this to $\mathscr{R}_1$. We shall also introduce semantic maps with special structure, see Notes A.

Definition 7.1. *A semantic map is called backward looking if any connection* $\sigma_{ij}(x)$ *connects all new out-bonds* (if any at all) *of generators introduced by it to generators already in* c, $c \in C_i$.

Similarly we could speak of forward looking semantic maps, but this notion will not be more than mentioned in this paper.

7.2.2. As an example of a backward looking semantic map the one in 6.3 can be mentioned. The only connectors $\sigma_{ij}(x)$ that introduce new generators in Figure 6.2 are $\sigma_{23}(a)$, $\sigma_{33}(g)$, $\sigma_{45}(b)$, $\sigma_{55}(d)$ and they all connect the new generators to old ones.

Lemma 1. *If the semantic map is backward looking we have* $\mathrm{sem}({}^1u^i) \in \mathscr{C}_1$ *for any grammatical phrase starting at state* 1.

Proof: Any semantic map, in the sense we use the term, automatically leads to local regularity for $\mathrm{sem}({}^1u^i)$. Indeed, $\mathrm{sem}({}^1u^i) \in C_i$. But the configurations in C_i belong to $\mathscr{C}_1$ so that all closed bonds satisfy ρ_1. Therefore it is only necessary to verify global regularity. But each connector in the semantic unwrapping of ${}^1u^i$ either does not contain any generator, or if it does, all their out-bonds are connected immediately. Therefore all the out-bonds of $\mathrm{sem}({}^1u^i)$ are closed and the subconfigurations introduced have the connector type Σ_1. In other words $\mathrm{sem}({}^1u^i)$ is $\mathscr{R}_1$-regular. Q.E.D.

7.2.3. Any grammatical phrase ${}^1u^i$ now means a regular configuration, an important fact that will facilitate the learning of the semantics. The reason for this is that, given a sentence $x_1x_2\ldots x_n \in L$, we can consider each initial phrase $u_k = x_1x_2\ldots x_k$, starting with small values of k, and attempt to learn the meaning of each new branch in the wiring diagram. This makes sense only if, as here, each ${}^1u^i_k$ is meaningful in $\mathscr{C}_1$, not just in $\mathscr{E}_1$ where the configurations do not always imply any meaning to the observer.

7.3.1. To build up a semantic category, see Theorem 6.1, we must construct the connectors $\sigma_{ij}(x)$, but so far we have only seen some simple examples of how this can be done.

To penetrate our problem deeper we shall use the concept of *bonding function which maps syntactic information* (from the sentence in $\mathscr{C}_2$) into *topological information* (for the perceived configuration in $\mathscr{C}_1$). We believe that this concept will be of fundamental importance in further work on mathematical semantics.

We first give the formal definition of a bonding function, and then illustrate its use by examples.

7.3.2. With $B = B_{in}$ = the set of bond values (for G_1) introduce the set

$$D = B \cup (B\times B) \cup (B\times B\times B) \cup \ldots \triangleq D_1 \cup D_2 \cup D_3 \cup \ldots \quad (7.2)$$

and a set Φ of functions ϕ defined on subsets of D. Denote by $D(\phi)$ the domain of such a *bonding function* ϕ, $D(\phi) \subseteq D$.

A bonding function will always be associated with a bond value $\beta \in B$, and we shall assume that for $\delta = (b_1, b_2, \ldots b_n) \in D_n$ the bonding function takes values in the set

$$\Delta_n(\beta) = \{i | b_i = \beta\}. \quad (7.3)$$

The purpose of the bonding function is to select one of the bonds of the generators introduced that have the in-bond value β. The set $\Delta_n(\beta)$ can consist of all the integers $1,2,\ldots n$. We shall make sure that no problem arises from the possibility $\Delta_n(\beta) = \phi$ by restricting the domain $D(\phi)$ appropriately.

7.3.3. Recall that the topology of $\mathscr{C}_1$-configurations typically looks as Figure 3.3 with the generators arranged in

layers of increasing level of abstraction. This makes it natural to attempt to organize the syntax $\mathscr{G}$ and the syntactic map sem in a similar way. Passing through the wiring diagram we would first handle the objects, level 0, then the properties, level 1, and connect them to the objects, and so on. The connections will be established by bonding functions attached to the branches of the wiring diagram.

Say that we have a branch $i \overset{x}{\to} j$ whose connector operates on level ℓ, $\ell \geq 1$. To this branch we associate at most one generator, say $g \in G_\ell^\ell$ and with $\omega_{out}(g) = \omega$, the out-bond values being $\beta_1, \beta_2, \ldots \beta_\omega$, as well as ω bonding functions $\phi_1, \phi_2, \ldots \phi_\omega$. Here ϕ_r should be associated with the bond value β_r. We allow the degenerate cases when a branch is associated with no generator, only bond functions, or with no generator and no bond function.

Then the connector $\phi_{ij}(x)$ should be formed by connecting the r^{th} out-bond of g to generator number $\phi_r(\delta)$ in the previous level ℓ-1. The vector $\delta = (\beta_1', \beta_2', \ldots \beta_n')$ describes the in-bond values of the subconfiguration consisting of the generators of level ℓ-1, enumerated in the order they have been generated.

In order that this make sense we must ensure that $\delta \in D(\phi)$ which will be done in the following by restricting the selection of any bonding function by what branches precede the current branch in the wiring diagram.

7.4.1. To make the above more intuitive consider the image algebra in 4.2 restricted to generators of levels 0 and 1. Choose L with $V_T = \{\alpha, \beta, \gamma, \delta\}$, and $V_N = \{1, 2, \ldots 10, 11, F\}$ with the wiring diagram in Figure 7.3.

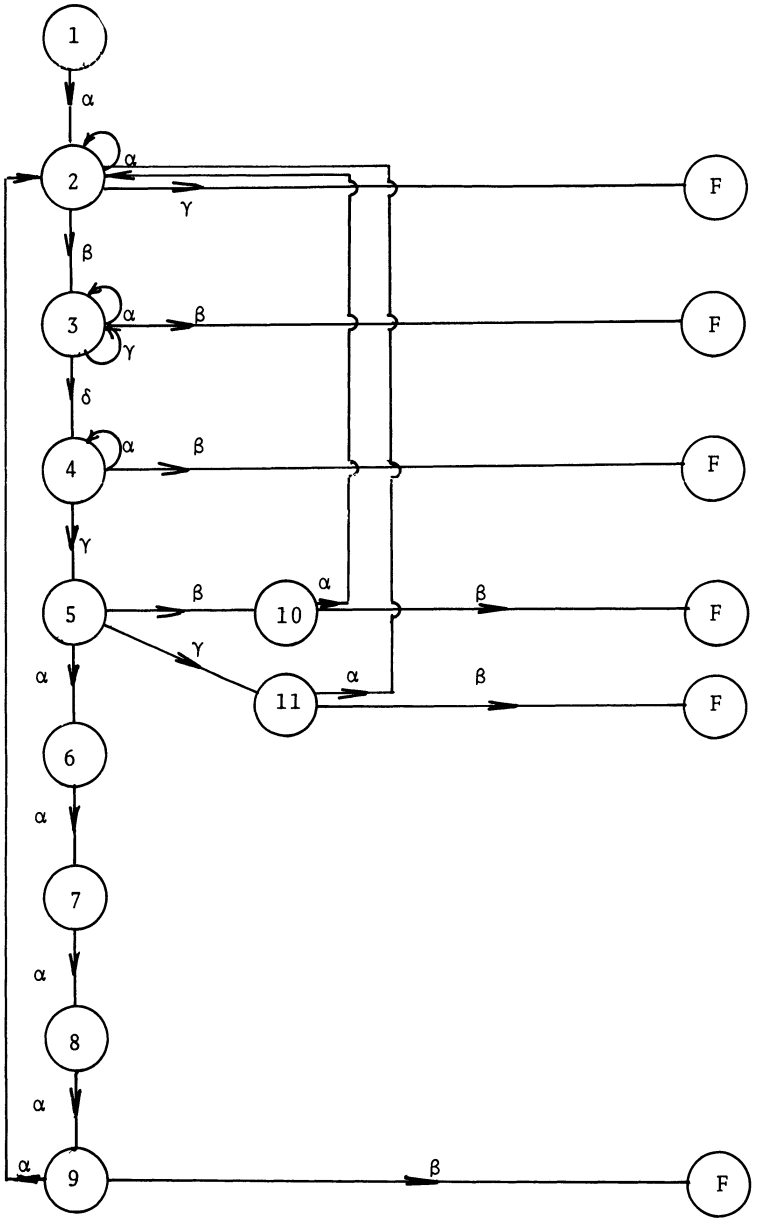


Figure 7.3

TABLE 7.1

branch $i \overset{x}{\rightarrow} j$	connector $\sigma_{ij}(x)$
$1 \overset{\alpha}{\rightarrow} 2$, $2 \overset{\alpha}{\rightarrow} 2$, $2 \overset{\beta}{\rightarrow} 3$ $9 \overset{\alpha}{\rightarrow} 2$	ϕ_1
$3 \overset{\delta}{\rightarrow} 4$, $3 \overset{\delta}{\rightarrow} 3$, $3 \overset{\gamma}{\rightarrow} 5$ $5 \overset{\alpha}{\rightarrow} 6$, $6 \overset{\alpha}{\rightarrow} 7$, $7 \overset{\alpha}{\rightarrow} 8$	ϕ_2
$8 \overset{\alpha}{\rightarrow} 9$, $9 \overset{\beta}{\rightarrow} F$	$\phi_3, \phi_4, \phi_5, \phi_6, \phi_7$
$5 \overset{\beta}{\rightarrow} 10$	$\phi_8, \phi_9, \phi_{10}$
$3 \overset{\alpha}{\rightarrow} 3$	ϕ_{11}
$3 \overset{\gamma}{\rightarrow} 3$	ϕ_{12}
$5 \overset{\gamma}{\rightarrow} 11$	$\phi_{13}, \phi_{14}, \phi_{15}, \phi_{16}$
all others	no change

To create a semantic map we shall use the bonding functions given in Table 7.2 and interpret the grammatical productions according to Table 7.1.

The "definition of ϕ" listed in the second column of Table 7.2 means the entire action of *all* the bonding functions listed in a row. The role of the individual bonding functions is given only implicitly. For example ϕ_3 means add an f and connect $b_{out,1}$ to the last k, while ϕ_4 means connect

TABLE 7.2

bonding function ϕ	definition of ϕ
ϕ_1	add unconnected ℓ
ϕ_2	add unconnected k
ϕ_i, i = 3,4,5,6,7	add f and connect $b_{out,i-2}$ to $(i-2)^{th}$ of last k
ϕ_i, i = 8,9,10	add g and connect b_{out1}, b_{out2} to last two k's and b_{out3} to last ℓ
ϕ_{11}	add h and connect to last ℓ
ϕ_{12}	add j and connect to last ℓ
ϕ_i, i = 13,14,15,16	add i and connect b_{out1} and b_{out2} to last two ℓ's and b_{out3} and b_{out4} to last two k's

$b_{out,2}$ (of the current f-generator) to the next last k.

Consider the sentence

$$I_2 = \alpha\beta\delta\gamma\beta\beta \tag{7.4}$$

with the parsing

$$1_\alpha 2_\beta 3_\delta 4_\gamma 5_\beta 10_\beta F. \tag{7.5}$$

It is grammatical.

To unwrap the meaning of I_2 we get by successively applying the connectors formed by using the bonding functions in the tables:

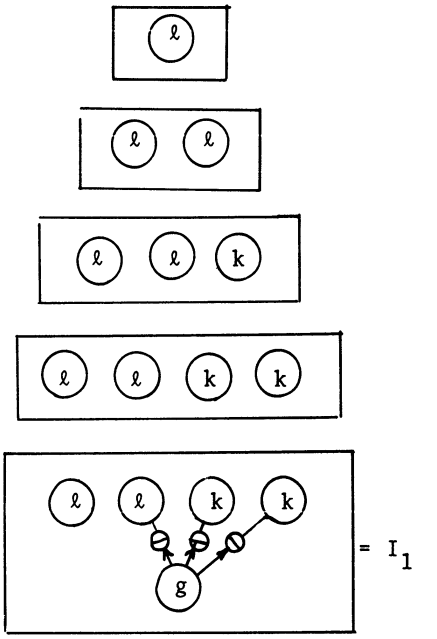
$1_\alpha 2$

$1_\alpha 2_\beta 3$

$1_\alpha 2_\beta 3_\delta 4$

$1_\alpha 2_\beta 3_\delta 4_\gamma 5$

$1_\alpha 2_\beta 3_\delta 4_\gamma 5_\beta 10$



The last transition $10 \to F$ does not change the image.

Now a more complicated example is

$$I_2 = \alpha\beta\delta\gamma\gamma\alpha\beta\delta\gamma\alpha\alpha\alpha\beta \tag{7.6}$$

parsed into

$$1_\alpha 2_\beta 3_\delta 4_\gamma 5_\gamma 11_\alpha 2_\nu 3_\delta 4_\gamma 5_\alpha 6_\alpha 7_\alpha 8_\alpha 9_\beta F. \tag{7.7}$$

Applying the same unwrapping procedure we see that $\underline{\text{sem}}(I_2) = I_1$ given in Figure 7.4.

Remark 1. The connectors used in the example have two properties that we will meet under more general conditions. Each bonding function connects out-bonds (if any at all) of new generators to in-bonds of old generators; the resulting semantic map is backward looking.

Remark 2. Any bonding function in Table 7.2 is defined in terms of the $1^{st}, 2^{nd}, \ldots$ of the last in-bonds with a given

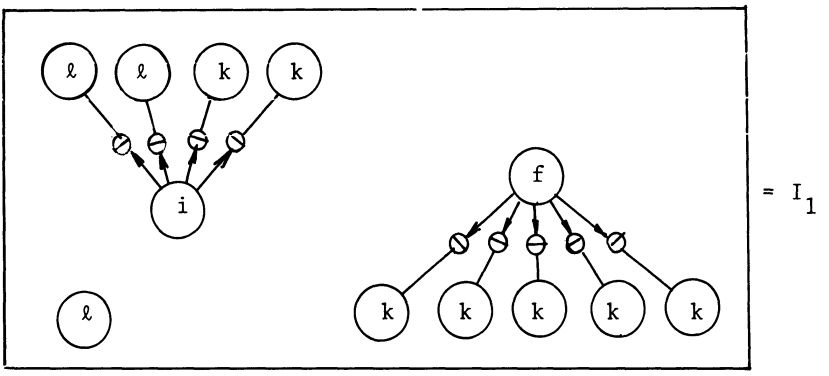


Figure 7.4

value. This may not be immediately obvious since Table 7.2 appears to mention certain generators rather than their in-bonds. Referring to Table 4.1, the last two rows, we see however that this amounts to the same thing in the present example. In other, more general cases, this distinction must be kept in mind. Such bonding functions, depending only upon the order in which generators and bonds have been introduced will be said to employ *ordered reference*.

7.4.2. Now return to the wiring diagram in Figure 7.3. A state can be identified with the set of strings $^{1}u^{i}$ leading from 1 to i. Actually, Nerode's theorem tells us that if we use as states the congruence classes over V_T^* we get the minimal wiring diagram.

A semantic map, given in terms of such bonding functions that were mentioned in the last two remarks, depends crucially upon the numbers

$$N_i(\beta) = \min\#\{g\text{'s introduced by any } {}^{1}u^{i} \text{ with } \beta_{in}(g) = \beta, \quad \beta \in B,\ i \in V_N\}. \tag{7.8}$$

In (7.8) the minimum is taken over all phrases starting in 1 and ending in i.

In our example we have

$$\begin{cases} N_1(\beta) = 0, \text{ all } \beta \\ N_2(I) = N_2(II) = N_2(III) = N_2(IV) = N_2(V) = 0,\ N_2(VI) = 1 \\ N_3(I) = N_2(II) = N_2(III) = N_2(IV) = N_2(V) = 0,\ N_2(VI) = 2 \\ \cdots \end{cases} \tag{7.9}$$

as can be verified going back to Table 4.1, fourth column.

The numbers $N_i(\beta)$ tell us how much topological information we have built up at state i expressed in terms of a lower bound for the number of potential in-bonds.

7.4.3. A related set of numbers are the *lags* $\lambda_\beta(\phi)$ of a bonding function ϕ employing ordered references. It means

$$\lambda_\beta(\phi) = \max\{\text{number of steps backwards of references to } \beta\text{-values for } \phi\} \tag{7.10}$$

In the example we have

$$\begin{cases} \lambda_\beta(\phi_1) = \lambda_\beta(\phi_2) = 0 \\ \lambda_V(\phi_i) = i-2,\ \lambda_\beta(\phi_i) = 0;\ i = 3,4,5,6,7 \\ \lambda_V(\phi_i) = i-7,\ \lambda_\beta(\phi_i) = 0 \text{ all other } \beta;\ i = 8,9 \\ \lambda_{VI}(\phi_{10}) = 3,\ \lambda_\beta(\phi_{10}) = 0 \text{ all other } \beta; \\ \cdots \end{cases} \tag{7.11}$$

The lag tells us how far back we have to remember potential in-bonds of generators that have already been unwrapped.

7.5. Leaving the example, consider now the connectors constructed as above. Does it lead to a semantic category as in Theorem 6.1? An answer is given by

Theorem 1. *Consider a backward looking strategy and assume that for any branch* $i \overset{\times}{\to} j$ *in the wiring diagram any associated bonding function* ϕ *with bond value* β *satisfies*

$$\lambda_\beta(\phi) \leq N_i(\beta). \tag{7.12}$$

Then the construction leads to an entire semantic map.

Proof: We construct the connectors $\sigma_{ij}(x)$ directly by executing the commands in the bonding functions ϕ belonging to the branch $i \overset{\times}{\to} j$. Each time we have zero or one generator whose out-bonds have to be connected. The bonding functions do this without ambiguity since only one generator is concerned as far as out-bonds go.

With the aid of the $\sigma_{ij}(x)$ connectors we can now build up the classes C_i, starting with the empty configuration at state 1 and connecting more generators or closing bonds as commanded by the $\sigma_{ij}(x)$. We have to make sure that these classes are subsets of $\mathscr{G}_1$; see Definition 6.1.

This is so; in the present case we can even assert that $C_i \subseteq \mathscr{L}_1 \subseteq \mathscr{G}_1$: all the configurations that we unwrap sequentially are regular. As for global regularity this follows from what was said in the proof of Lemma 7.1.

Local regularity does not follow quite as directly. Indeed, it could happen when we build up the classes C_i that the value of a connector, when applied to the current configuration, is not defined. But such a connector is made up by bonding functions, each ϕ of which only refers backwards a certain number of steps in the order of reference. If fewer generators with the relevant in-bonds had been introduced so far the procedure would fail. Condition (7.12) insures

that this cannot happen: we have access to the required number of relevant in-bonds in our list of potential ones. Therefore ϕ is always defined, the bonds can be closed without violating the bond relation ρ, and the new configuration will be regular. Q.E.D.

7.6. It is obvious how a *forward looking strategy* would be organized. This will not be examined in detail here, but we shall have occasion to study strategies looking both forward, for some bonding functions, and backward for others. Theorem 7.1 will then have to be modified.

Since lags can then be both positive and negative we also need a function given as the maximum of the absolute value of the negative lags involved; we will use both $\lambda_\beta^+(\phi)$ as before and the new $\lambda_\beta^-(\phi)$.

We also need an analogue of the numbers $N_i(\beta)$ in (7.8) and introduce

$$M_i(\beta) = \min \#\{g' \text{ introduced by any } {}^{i}_{u}F \text{ with } \beta_{in}(g) = \beta\};\ \beta \in B,\ j \in V_N. \tag{7.13}$$

There will now be two conditions

$$\begin{cases} \lambda_\beta^+(\phi) \leq N_i(\beta) \\ \lambda_\beta^-(\phi) \leq M_i(\beta) \end{cases} \tag{7.14}$$

in order that our construction yield a semantic category.

Note that we can no longer assert that $C_i \subseteq \mathscr{C}_1$, only that $C_i \subseteq \bar{\mathscr{C}}_1$.

9.8. Learning semantics

Assuming the semantic map to be fixed but unknown, of backward looking type, and expressed as in the previous section,

$$\text{sem} = \{\sigma_{ij}(x)\} \tag{8.1}$$

how can it be learnt from finite experience? More precisely how can sem be estimated when we have observed a sequence of *image couples*

$$[I_1(t), I_2(t)];\ t=1,2,\ldots N;\ I_1(t) \in \mathscr{I}_1;\ I_2(t) \in \mathscr{I}_2 \tag{8.2}$$

where $\text{sem}[I_2(t)] = I_1(t)$. The sequence represents our finite experience and we have left out syntactically correct but meaningless sentences that may have been encountered.

Each $I_1(t)$ can be expressed by listing the generators and connections in one of the configurations that make up the image. Similarly each sentence $I_2(t)$ can be expressed by its passing into a sequence of productions $i \overset{x}{\to} j$. We are looking for a map from the set of productions to the set of generator - connectors.

The set of productions is given already and finite. The set of primitive connectors can be infinite. We shall bound it by assuming the maximum lag to be finite. We then are considering mappings between two finite sets. Consider two finite sets X and Y with $\#(X) = n$ and $\#(Y) = m$. An unknown map $\alpha: X \to Y$ should be estimated when we have access to the observed set couples, $x(t) \subseteq X$, $y(t) \subseteq Y$,

$$\begin{cases} x(1), y(1) = \alpha x(1) \\ x(2), y(2) = \alpha x(a) \\ \vdots \\ x(N), y(N) = \alpha x(N) \end{cases} \tag{8.3}$$

Of course αx stands for the set of all y-values obtained from the point-to-point map α applied to all elements of x.

We have not said anything about the way the x(t)-sequence has been generated. In the following it will be assumed that it is obtained as an i.i.d. sample with respect to a given probability measure over 2^X. The measure could be for example one of the syntax controlled measures for context free languages studied in Volume I, or some other measure of interest.

It is important to realize that our problem has been reduced to estimating a *function, here denoted by* α, *when only sets can be observed.* This important problem does not seem to have been studied in the statistical literature, so that we have to start by developing methods for doing this.

Before doing this let us just mention the following possible extensions of our problem. When we have the parsing of an image for $\mathscr{T}_2$ it consists of a string of productions. We abstract the information of set type from this string: the set of productions that occur in the string. Of course this destroys information. The string (1,2,2,7,14,11,7,2) is reduced to the set {1,2,7,11,14}.

We could preserve more information if we also counted frequencies of occurences in the string. In the example we would not just say that "2" occurred, but that it occurred twice, and so on. In other words we would be treating x(t) in (8.3) as a multi-set, not just a set, and similarly for y(t).

Still better would be to treat the string as ordered, then we would lose no information. Similarly treat the

generator-connector occurences in $\mathscr{L}_1(\mathscr{R})$ as a POSET.

We are not ready to deal with these two more informative versions of the problem and hope that other researchers will do this.

Returning to the first version, how can we estimate α? It will be instructive to do this first assuming that

$$\alpha \text{ is bijective (so that } n = m). \tag{8.4}$$

We shall later see what happens under more general conditions. We can immediately make a simple statement

Lemma 1. *The observations imply that*

$$\alpha(\delta_i) \in D_{i,N} = \left[\bigcap_{x_i(t)=1} y(t)\right] \cap \left[\bigcap_{x_i(t)=0} y^c(t)\right]; \ \forall i; \tag{8.5}$$

where δ_i *consists of the* i^{th} *element of* X *and* $x(x_1(t),x_2(t),\ldots x_n(t))$ *is written as an indicator function. This is the strongest possible statement.*

Proof: Consider the set

$$E_{i,N} = \left[\bigcap_{x_i(t)=1} x(t)\right] \cap \left[\bigcap_{x_i(t)=0} x^c(t)\right]. \tag{8.6}$$

It is clear that $\delta_i \in E_{i,N}$ since both factors in (8.6) have the i^{th} element in it for all t. But then $\alpha(\delta_i) \in \alpha(E_{i,N})$, and using the fact that α is *homomorphic* for set operations since it is bijective, we get $\alpha(E_{i,N}) = D_{i,N}$.

We cannot strengthen (8.5). Indeed, let the map α satisfy (8.5). If $\#(D_{i,N}) = 1$ then the value $\alpha(\delta_i)$ is completely determined. If $\#(D_{i,N}) > 1$ then $D_{i,N}$ contains two distinct elements, say k and ℓ, and with $\alpha(\delta_i) = k$. We now claim that two $E_{i,N}$ sets are either equal or disjoint.

We can write the intersection of $E_{i,N}$ and $E_{j,N}$ as

$$E_{i,N} \cap E_{j,N} = \left[\bigcap_A x(t)\right] \cap \left[\bigcap_B x^c(t)\right] \tag{8.7}$$

where

$$\begin{cases} A = \{t | x_i(t)=1 \text{ or } x_j(t)=1\} \subseteq \{1,2,\ldots n\} \\ B = \{t | x_i(t)=0 \text{ or } x_j(t)=0\} \subseteq \{1,2,\ldots n\} \end{cases} \tag{8.8}$$

It is clear that if A and B have an element in common then the intersection in (8.7) is empty. But if A and B exclude each other, since $A \cup B = \{1,2,\ldots n\} = n$, we must have $A = B^c$. This means that

$$[x_i(t)=1 \text{ or } x_j(t)=1] \iff [x_i(t)=1 \text{ and } x_j(t)=1]. \tag{8.9}$$

Then we must have, for a given t, either $x_i(t) = x_j(t) = 0$, or one of them is $= 1$. In the letter case (8.9) implies that both are equal to 1. Hence $x_i(t) = x_j(t)$, $\forall t$. Recalling the definition (8.6) we see that $E_{i,N} = E_{j,N}$. Hence $\{E_{i,N}\}$ constitutes a partition of X. Introduce a new map α' which is equal to α except that it permutes k and ℓ. Note that α maps the sets $E_{i,N}$ to $D_{i,N}$ so that the construction is possible. But α' is consistent with the observations; hence (8.5) is the strongest possible statement based on the observations. Q.E.D.

Remark 1. The proof uses the homorphic property several times, and this is based on the assumption that α is bijective.

Given a sequence of sets we shall say that it is *complete* if the resulting $E_{i,N} = \delta_i$, $\forall i$.

Theorem 1. *As* $N \to \infty$ *we will learn* α *consistently (with probability* 1*) if and only if* support (P) *forms a complete set.*

Proof: Assume that support (P) is complete. Then, with probability one, the infinite i.i.d. sample will contain each of the elements in the support. Then Lemma 1 gives us α unambiguously.

On the other hand in order that $E_{i,N} \to \delta_i$, $\forall i$, as $N \to \infty$ with probability one (which is needed according to the indirect part of the lemma) we can argue as follows. As N increases we will get more and more factors in the intersection on the right side of (8.6). The sequence of $E_{i,N}$ sets is monotonic and hence has a limit. This limit is a.c., with $S = \text{support}(P)$,

$$E_{i,\infty} = \left[\bigcap_{\substack{x_i=1 \\ \underline{x} \in S}} x \right] \cap \left[\bigcap_{\substack{x_i=0 \\ \underline{x} \in S}} x^c \right]. \tag{8.10}$$

But $E_{i,\infty} = \delta_i$, $\forall i$, if and only if S forms a complete set, as asserted. Q.E.D.

Given a set S we can give an algorithm for determining whether it is complete or not. This could be done similarly by Gram-Schmidt orthogonalization (sequentially), or directly.

One case in which P is obviously complete is when $\text{support}(P) = 2^X$: all sets occur with positive probability.

We now turn to the somewhat more complicated question: what is the speed of learning? To ensure completeness say that we are in the situation just mentioned when $\text{support}(P) = X$. For example, all sets x could be given the same probability 2^{-n}: the uniform distribution over X. An answer is given by the following result; see Notes A.

Theorem 2. *If* $\text{support}(P) = 2^X$ *then the probability that we will have learnt* α *completely after* N *trials satisfies*

$$\lim_{N\to\infty} \frac{1\text{-P(complete learning)}}{\nu\mu^N} = 1 \tag{8.11}$$

where

$$\mu = \max_{i<j} P(x_i = x_j) \tag{8.12}$$

and ν *is the number of pairs* $i < j$ *realizing the maximum.*

Proof: To determine the probability that $x(1),x(2),\ldots x(N)$ is complete, $E_{i,N} = \delta_i$, $\forall i$, introduce for $i < j$ the event F_{ij} with indicator function

$$\prod_{x_i(t)=1} x_j(t) \cdot \prod_{x_i(t)=0} [1-x_j(t)]. \tag{8.13}$$

Then the event F = "the system is complete" can be expressed as

$$F = \bigcap_{i<j} F_{ij}^c; \quad F^c = \bigcup_{i<j} F_{ij}. \tag{8.14}$$

But then

$$P(F^c) = \sum_{i<j} P(F_{ij}) - \sum_{\substack{i<j \\ k<\ell}} P(F_{ij} \cap F_{k\ell}) + - \ldots \tag{8.15}$$

where we have ordered all pairs (i,j), $i < j$, and consider only $(i,j) < (k,\ell)$ etc. in the above sums.

We have, $i < j$, using the i.i.d. property

$$P(F_{ij}) = E\left\{ \prod_{x_i(t)=1} x_j(t) \prod_{x_i(t)=0} [1-x_j(t)] \right\} = c_{ij}^N \tag{8.16}$$

where

$$\begin{aligned} c_{ij} &= P\{x_i=1 \text{ and } x_j=1 \text{ or } x_i=0 \text{ and } x_j=0\} \\ &= P\{x_i=x_j\}. \end{aligned} \tag{8.17}$$

Hence the terms in the first sum of (8.15) that matter asyptotically give together the contribution $\nu\mu^N$, and it remains to show that the rest is negligible.

Consider a term in (8.15) of the form

$$P(F_{ij} \cap F_{k\ell}) \tag{8.18}$$

where $(i,j) \neq (k,\ell)$. First, assume that all four subscripts are different. Then the probability (8.18) is

$$E\left\{ \prod_{x_i(t)=1} x_j(t) \prod_{x_i(t)=0} [1-x_j(t)] \prod_{x_k(t)=1} x_\ell(t) \prod_{x_k(t)=0} [1-x_\ell(t)] \right\}$$

$$= c^N_{ij;k\ell} \tag{8.19}$$

where

$$\begin{aligned} c_{ij,k\ell} &= P\{x_i=x_j \text{ and } x_k=x_\ell\} \\ &= c_{ij} P\{A|B\} \end{aligned} \tag{8.20}$$

with

$$\begin{cases} A = \{x_k = x_\ell\} \\ B = \{x_i = x_j\}. \end{cases} \tag{8.21}$$

If $P(A|B) = 1$ we have $P(A \cap B) = P(B)$ and $P[B \cap (A \cap B)^c] = 0$ so that $P[B^c \cup (A \cap B)] = 1$. But

$$(x_i \neq x_j) \vee [(x_k=x_\ell) \wedge (x_i=x_j)] \tag{8.22}$$

is not a tautology; it is false, for example when $x_i=x_j=0$, $x_k=0$, $x_\ell=1$. Hence $P(A|B) < 1$ and

$$c_{ij,k\ell} < c_{ij} \tag{8.23}$$

and the corresponding terms in (8.15) are asymptotically negligible.

Second, assume that $i = k$ differs from j and from $\ell \neq j$. Then (8.19) should be replaced by

$$E\left\{ \prod_{x_i(t)=1} x_j(t)x_\ell(t) \prod_{x_i(t)=0} [1-x_j(t)][1-x_\ell(t)] \right\} = d^N_{i,j\ell} \tag{8.24}$$

where

$$d_{i,j\ell} = P\{x_i=x_j=x_\ell\} = P(C|B)c_{ij} \tag{8.25}$$

with

$$C = \{x_\ell = x_j\}. \tag{8.26}$$

The same reasoning as above leads now to the Boolean function

$$(x_i \neq x_j) \vee [(x_i = x_j) \wedge (x_\ell = x_j)] \tag{8.27}$$

which is false, for example when $x_i = x_j = 0$, $x_\ell = 1$.

The remaining terms are treated in the same way. Q.E.D.

Remark 1. Given a level $\varepsilon > 0$ we can expect to have a complete system of observations if N is at least (note that $\ell n\ \mu < 0$!)

$$N_\varepsilon = \frac{\ell n\ \varepsilon - \ell n\ \nu}{\ell n\ \mu} \quad . \tag{8.28}$$

In the special case of P uniform over $X = 2^m$ we have $P(x_i = x_j) = 1/2$ for all $i < j$ so that $\mu = 1/2$ and $\nu = n(n-1)/2$. With base 2 log's in (8.28) we get

$$N_\varepsilon = \log_2 \frac{n(n-1)}{2} - \log_2 \varepsilon \sim 2 \log_2 n \text{ for large } n \tag{8.29}$$

where the factor 2 seems counter-intuitive! (Compare with the lower bound $\log_2 n$.)

In a small mathematical experiment P was made uniform and sampling over 2^X was continued until a complete system was obtained. An APL program was written to test for completeness. The following sample sizes T were observed

n	T	n	T
4	8	16	7
4	7	16	7
4	8	16	13
4	4	32	9
8	5	32	7
8	8	32	13
		64	12

The behavior predicted by (8.29) seems to apply fairly well for the larger values of n.

Remark 2. The leading term $\nu\mu^N$ is not stable. Indeed we can change P by an arbitrary small amount such that μ is not changed but ν changes by ± 1.

Remark 3. The expression in (8.11) makes it natural to define the *asymptotic learning rate* by

$$R = \overline{\lim_{N\to\infty}} P^{-1/N} \quad \text{(not complete learning)}. \tag{8.30}$$

In the present case $R = 1/\mu$. The worst cases occur when μ in (8.12) is large so that there is some pair (i,j) for which $P(x_i=x_j)$ is close to 1. Fast learning occurs if all $P(x_i=x_j)$ are small, and the question arises what the smallest value of $\mu = \mu(P)$ is when n is fixed. This is less obvious then the opposite extreme case, and is answered by

Theorem 3. *For given* n *we have* (with brackets indicating integral part) *and the minimum taken over all* P

$$\min \mu(P) = \frac{[\frac{(n-1)^2}{2}]}{n(n-1)} . \tag{8.31}$$

Proof: Introduce the indicator function

$$d_i \triangleq d_i(x) = 1_{x_i=1}(x) \tag{8.32}$$

and note that

$$r_{ij} = P(x_i=x_j) = E[1-(d_i-d_j)^2] \tag{8.33}$$

so that

$$A_n = \sum_{i\neq j} r_{ij} = E(B_n); \quad B_n = \sum_{i\neq j} [1-(d_i-d_j)^2]. \tag{8.34}$$

Expanding the square in (8.34) and noting that $d_i^2 = d_i$ we

can write

$$\begin{aligned} B_n &= \sum_{i\neq j} [1-d_i^2-d_j^2 + 2d_i d_j] \\ &= n(n-1)-2(n-1)\sum_i d_i^2 + 2\sum_i d_i \sum_{j\neq i} d_j \\ &= n(n-1)-2(n-1)\sum_i d_i + 2\sum_i d_i\Big(\sum_j d_j - d_i\Big) \\ &= n(n-1)-2n\,D + 2D^2 \end{aligned} \tag{8.35}$$

with $D = \Sigma d_i$. This quadratic polynomial attains its minimum for $D_o = n/2$.

Now D is a stochastic variable taking as possible values $0,1,2,\ldots n$. If n is even, $n = 2m$, then the inequality follows by direct calculation (again with brackets indicating integral parts)

$$B_n \geq \frac{n^2}{2} - n = 2m^2-2m = [\frac{n^2}{2} - n + \frac{1}{2}] = [\frac{(n-1)^2}{2}] \ . \tag{8.36}$$

If n is odd, $n = 2m+1$, we get instead, since D can take integral values only,

$$B_n \geq \frac{n^2}{2} - n + \frac{1}{2} = \frac{(n-1)^2}{2} \ . \tag{8.37}$$

Hence (8.36) and (8.37) yield

$$A_n = E(B_n) \geq [\frac{(n-1)^2}{2}] \ . \tag{8.38}$$

But

$$A_n = \sum_{i\neq j} r_{ij} \leq n(n-1) \max_{r\neq j} r_{ij} = n(n-1)\mu \tag{8.39}$$

so that, as claimed,

$$\mu \geq \frac{[\frac{(n-1)^2}{2}]}{n(n-1)} \ . \tag{8.40}$$

It is clear from the method of the proof that in order that equality hold in (8.40) we must have, for even n,

$$P[D(x) = \frac{n}{2}] = 1 \tag{8.41}$$

and, for odd n,

$$P[D(x) = \frac{n-1}{2} \text{ or } \frac{n+1}{2}] = 1. \tag{8.42}$$

To show that the bound (8.40) is achieved pick a vector $\xi = (1,1,1,\ldots 0,0)$ where the number of 1's is $n/2$ if n is even, and $(n+1)/2$ if n is odd. Apply all permutations to ξ and define P by having the same probability $1/n!$ for each one. Then P is symmetric so that all r_{ij}, $i \neq j$, are equal. But this common value r must then satisfy, since $D(x)$ satisfies (8.41) or (8.42),

$$A_n = n(n-1)r = [\frac{(n-1)^2}{2}] \tag{8.43}$$

so that the bound (8.40) is achieved. Q.E.D.

The estimate of α given in (8.5) will be called the *intersection algorithm* when we want to distinguish it from others to be introduced later. Theorem 1 tells us that if the support of P forms a complete set the intersection algorithm is guaranteed to converge to the correct map. Furthermore, we know the asymptotic speed of convergence given by Theorem 2, and the asymptotic learning rate equals $1/\mu$ with μ defined in Eq. (8.12). This implies that unless P is very skewed over 2^X we can expect to find the map α quickly. It is only when P is quite skew that we need to expect trouble. An extreme case occurs when P attributes probability zero to many subsets of X so that the support is not complete: Then we cannot hope to get convergence to the true α.

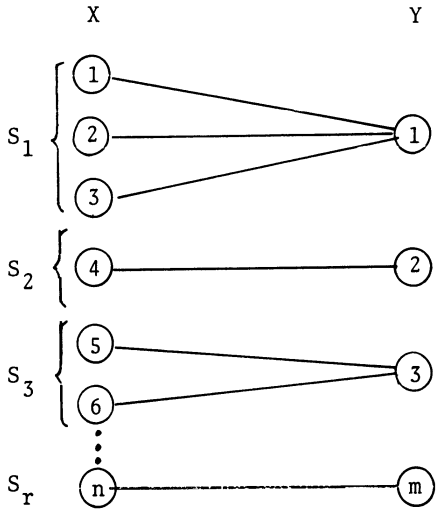


Figure 8.1

An estimate of quite different type has been suggested by S. Geman. This *random algorithm* starts at iteration $t = 1$ with an arbitrary assignment to α, call it $\alpha^*(1)$. When we observe the set couple $[x(t),y(t)]$ we do not change any component of $\alpha(t)$ that is consistent with our information. A component that is not consistent is changed at random, with equal probability to the consistent possibilities.

We shall study the asymptotic learning behavior of the random algorithm. We shall no longer assume that it is bijective - only surjective. The α-map therefore looks typically as in Figure 8.1 after a relabelling of the x- and y-values.

Let the set S_k that is mapped by α into $y = k$ be of size n_k. If all $n_k = 1$, α is bijective. Consider one of the components of $\alpha^*(t)$, for example the first one. Denote it by $a(t)$ so that we would like to assert that $a(t) \to 1$ in the case illustrated in the figure. It is easy

to verify that $a(t)$ forms a Markov chain with m states and some matrix Q of transition probabilities looking as

$$Q = \begin{Bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ q_{21} & q_{22} & q_{23} & q_{24} & \cdots & q_{2m} \\ \vdots & & & & & \\ q_{m1} & q_{m2} & q_{m3} & q_{m4} & \cdots & q_{mm} \end{Bmatrix} \qquad (8.44)$$

where the $(m-1) \times (m-1)$ matrix in the box will be denoted R. The fact that the first row of Q looks as in (8.44) is a direct consequence of the fact that once $a(t)$ has attained the value 1 no information can ever contradict this since it is the true value; therefore $a(t)$ will not change any further.

The probability at iteration t that the first component of the map has not been correctly identified is then

$$P[a(t) \neq 1] = p^T Q^t e = \sum_{k=2}^{m} p_k \sum_{\ell=2}^{m} Q_{k\ell}^t \qquad (8.45)$$

where $Q_{k\ell}^t$ stands for the (k,ℓ) element of the t^{th} power of Q, p is the m-vector of initial probabilities, and e is the vector $(0,1,1,\ldots 1)$. Note that only (k,ℓ) values belonging to the matrix R appear in the sum in (8.45).

To get an asymptotic bound for (8.45), consider q_{i1} for $2 \leq i \leq m$. This means the conditional probability

$$q_{i1} = P[a(t+1) = 1 \,|\, a(t) = i] \qquad (8.46)$$

and in order that the event $a(t+1) = 1$ in (8.46) takes place when $a(t) = i$ we must have

$$\begin{cases} i \notin y(t) = \alpha x(t) \\ \text{and happen to pick 1 at random from the set } \alpha x \end{cases} \qquad (8.47)$$

To find the probability that (8.47) occurs see Figure 8.1 and let for example $i = 2$. To get information leading us to change $a(t) = 2$ to $a(t+1) = 1$ we must have an occurence of $x = 1$ but no occurences of $x = 4$ (x in S_2). We will then pick 1 with probability $(1+e_3+e_4+\ldots)^{-1}$ if e_k stands for the indicator function of at least one $x \in S_k$ occurring. This is so since we then pick the new value with equal probabilities from a set of size $1+e_3+e_4+\ldots$. Combining these probabilities we get

$$q_{i1} = \frac{1}{2}\left(\frac{1}{2}\right)^{n_i} E\left(\frac{1}{1+e_3+e_4+\ldots}\right) . \tag{8.48}$$

Using the inequality between the harmonic and arithmetic means (8.48) leads to the inequality

$$q_{i1} \geq \frac{2^{-1-n_i}}{E(1+e_3+e_4+\ldots)} \geq \frac{2^{-1-n_i}}{m-1} . \tag{8.49}$$

It is easy to sharpen the inequality but we shall not do this here. Instead we note that (8.49) implies

$$\|R\| \leq \max_{2\leq i\leq m} \sum_{j=2}^{m} p_{ij} = 1 - \min_{2\leq i\leq m} q_{i1} \leq 1 - \frac{2^{-1-\nu}}{m-1} \tag{8.50}$$

with $\nu = \max n_i$, leading to the crude bound,

$$\|R\| \leq 1 - \frac{1}{2^{\nu+1}(m-1)} . \tag{8.51}$$

Going back to (8.45) we see that the error probability for the first component is bounded geometrically as the number of iterations increases

$$P\left[a(t) \text{ takes the wrong value}\right] = 0\left[\left(1 - \frac{1}{2^{\nu+1}(m-1)}\right)^t\right]. \tag{8.52}$$

As an example, if α is again bijective, $\nu = 1$, we see that for large $m = n$ it is enough to let t be considerably

larger than n in order to be sure about the convergence for any given component of the estimate map $\alpha^*(t)$. While the random algorithm learns more slowly than the intersection algorithm, its behavior is quite acceptable for learning bijective maps. If α is highly non-injective, however, so that ν is large, the behavior deteriorates.

In terms of the semantic map lack of injectivity means that several productions in the language correspond to the same bonding function in $\mathscr{G}_1(\mathscr{R})$. If $\mathscr{G}_1(\mathscr{R})$ involves generators with high abstraction levels we have found (only empirically) that it often happens that many productions mean the same bonding function: α is far from bijective; see Notes B. This will then lead to slower abduction of the semantic map.

9.9. Abduction of semantic maps

The two algorithms studied in the previous section can now be applied to estimate sem. We can of course not expect that the map between the semantic category and that of connectors be bijective. It may be thought that the intersection algorithm is still optimal in the sense that no stronger statement can be made than the one in Lemma 8.1. This was so in the bijective case. But one can show by examples that this is not so in general. Nevertheless it is believed that the intersection algorithm performs moderately well even in the non-bijective case.

Computer experiments have been performed by P. Flanagan (1980) in order to study how the semantic map can be learned. The results will be reported elsewhere.