ALGEBRA AND LANGUAGE THEORY

P. M. COHN

1. *Introduction*

In the early days of high-speed computers there was a hope that it would be possible to program a computer to translate from one language to another, and this led to an intensive study of language structure. The result has been disappointing in that we are still far from making translations by computer, but that is no cause for despair. In the first place it shows that we have underestimated the richness and intricacy of our natural languages, even when used for quite prosaic ends. Secondly, this study of language has led to mathematical models of languages which admittedly are too simple to reflect all the complexities of a natural language like English, but —quite apart from their intrinsic interest—these models are found to give a good approximation to certain programming languages. And thirdly, there has been a vigorous interaction with some parts of noncommutative algebra, with benefit to " mathematical linguist " and algebraist alike.

There is an extensive literature in the field, much of it written from the point of view of automata theory, but overlapping with orthodox linguistics, communication theory, probability, logic, and even algebra. Likewise there are several excellent expositions; we mention in particular Booth [**67**], Arbib [**69**], Hopcroft–Ullman [**69**], Gross–Lentin [**70**], Claus [**71**], and for an introduction which emphasizes the links with natural languages, Gross [**72**]. But many of these books are written for non-mathematicians, spending pages of explanation on points that a mathematical audience would take for granted, and nearly all concentrate on a particular aspect, depending on their bias. Since the subject is mathematical at heart, and perhaps not as widely known among mathematicians as it deserves to be, it seemed worthwhile to present a brief survey of the main results. An advantage in writing such a survey is that the author may skip long or technical proofs; but in this field even quite technical results often have an intuitive content, more readily appreciated in a sketch than a rigorous presentation, and I have tried to include such sketch proofs where possible. Like other accounts, this one has a bias, not unnaturally towards algebra. One of my aims has been to make clear the connexion with power series rings, and to include results on the latter which may be relevant to the study of language theory.

The methods used to study languages fall into three classes: (i) grammars, (ii) machines and (iii) algebraic methods. Of these, (i) and (ii) are discussed in §§2 and 3 respectively, and (iii) in §5 (monoids) and §6 (power series). §4 deals with transductions, really a formalization of the idea of translation, and §7 looks briefly at the attempts to relate the existing models to natural languages.

This article is based on a lecture given to the London Mathematical Society, on

Received 16 July 1974.

[BULL. LONDON MATH. SOC., 7 (1975), 1–29]

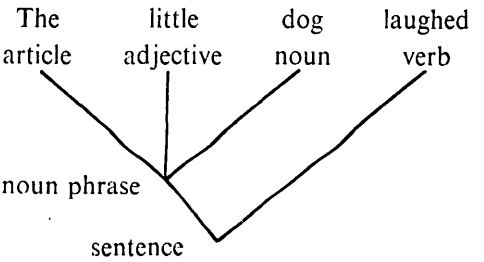
A

15th November, 1973. I am grateful to Professor D. G. Kendall, the President, for encouraging me to write it up. My thanks are also due to Professor G. M. Bergman and Dr. M. Fliess for correcting some inaccuracies and providing additional references.

2. *Grammars*

(i). Let X be a finite set, our *alphabet*. The set of all *words*, or strings of letters from X, is written X^*; it can be thought of as the free monoid (semigroup with 1) on X, with juxtaposition as multiplication, and including 1 to represent the empty word. The number of letters in a word u is called its *length*, written $l(u)$. By a *language* on X we understand any subset of X^*. We make no distinction between words and sentences; e.g. we could take one symbol x_0 of X as a blank space.

The first method of singling out interesting languages is by means of a grammar. Traditionally, grammar is used to parse a sentence, e.g.



We shall use grammar to build up a sentence, thus a grammar for us is a set of rules of the form: sentence $\rightarrow$ {noun phrase, verb}, noun $\rightarrow$ dog etc. which will lead us to the sentences of the language and no others. This amounts to reading the above diagram (also called a *derivation tree*) from the bottom upwards.

Formally we shall define a *phrase structure grammar* G to consist of

(i) an alphabet X; its letters are also called *terminal letters*;

(ii) a set V of *clause-indicators* or *variables*, also called *non-terminal letters*, including a symbol σ for a complete sentence. We write $A = X \cup V$.

(iii) a finite set of *rewriting rules*: $u \rightarrow v$, where $u, v \in A^*$ and u contains at least one variable.

A string of letters from A is called *terminal* if it belongs to X^*, *non-terminal* otherwise. We apply the rewriting rule $u \rightarrow v$ by replacing a string fug in A by fvg. To get a sentence f of the language we start from σ and apply the rewriting rules until no variables (clause-indicators) are left. The sequence of rules used constitutes a *derivation* of f; we also write $\sigma \rightarrow f$. In this way we obtain the language $L(G)$ generated by the given grammar G; it consists of the strings on X that can be reached by a derivation from σ, and no others.

Any language generated by a phrase structure grammar is called a *phrase structure language* or is said to be of *type* 0. It is not hard to see that the languages of type 0 are precisely the recursively enumerable subsets of X^*, so the study of these languages is just a part of recursive function theory, and further restrictions are needed to impart a typical linguistic flavour. This is achieved by imposing restrictions on the rewriting rules. Before giving examples of languages, we list the most widely studied types of grammars.

Type 1. *Context-sensitive* or *-dependent* grammars. All rewriting rules have the form

$$f\alpha g \to fug, \quad \text{where} \quad \alpha \in V,\ u \in A^*,\ u \neq 1.$$

This rule can be interpreted to mean: α is replaced by u *in the context* f_g. A grammar of this form is called a *CS-grammar* (or of type 1) and the language generated a *CS-language*.

Type 2. *Context-free* (Chomsky). All rewriting rules have the form

$$\alpha \to u, \quad \text{where} \quad \alpha \in V,\ u \in A^*,\ u \neq 1.$$

This means that α is replaced by u independently of the context. This is called a *CF-grammar* (or of type 2), and it generates a *CF-language*.

Type 3. *Regular* or *finite-state* (Kleene†). The rewriting rules have the form

$$\alpha \to \beta x, \quad \text{where} \quad x \in X,\ \alpha \in V,\ \beta \in V \text{ or } \beta = 1. \tag{1}$$

Here the special feature is that each variable is replaced either by a letter or by a variable followed by a letter. We speak of a *regular grammar* or a *K-grammar* and *-language*. Instead of writing the variable β on the left of the terminal letter in (1) we could also restrict our rules so as to have β on the right of the terminal letter throughout. It can be shown that this leads to the same class of grammars.

(ii). If $\mathscr{L}_i$ denotes the class of languages of type i ($i = 0, 1, 2, 3$), then it is clear that

$$\mathscr{L}_0 \supseteq \mathscr{L}_1 \supseteq \mathscr{L}_2 \supseteq \mathscr{L}_3,$$

and here all the inclusions can be shown to be strict, but in general it is not easy to tell where a given language belongs, since there will usually be many grammars generating it.

From the above definition it follows that no CS-language can include the empty word. This is often inconvenient; in fact a minor modification of the rewriting rules will allow us to include the empty word, if desired. We shall not enter into the details, but in what follows we shall feel free to include rewriting rules that lead to the empty word.

---

† These grammars were studied by Kleene **[56]** to obtain a model of the neuron.

Example 1. $\sigma \to x$, $\sigma \to \sigma y$. Let us derive a typical sentence:

$$\sigma \to \sigma y \to \sigma y^2 \to xy^2.$$

The complete set of sentences is $x, xy, xy^2, \ldots$, or briefly xy^*. Each sentence has just one derivation; we call such a language *unambiguous*. Clearly this is a regular language; here is a K-grammar generating the same language, but with the variables on the right:

$$\sigma \to x\lambda,\ \lambda \to y\lambda,\ \lambda \to 1.$$

Example 2. $\{x^n y^n \mid n \geqslant 0\}$. This is generated by the grammar

$$\sigma \to x\sigma y,\ \sigma \to 1.$$

This is a CF-grammar, so we have a CF-language. As it turns out, the language is not regular, but to show this we have to check, not merely that the above grammar is not regular, but that there exists no regular grammar generating the language. We shall reach this conclusion later by an indirect method.

Example 3. $\{x^n z^m y^n \mid m, n \geqslant 0\}$. This is again context-free but not regular. It is generated by the grammar

$$\sigma \to x\sigma y,\ \sigma \to \lambda,\ \lambda \to z\lambda,\ \lambda \to 1.$$

Here we cannot do without the variable λ; e.g. $\sigma \to x\sigma y$, $\sigma \to z\sigma$, $\sigma \to 1$ would also give zxy.

Example 4. $\{x^m y x^n \mid m \geqslant n \geqslant 0\}$. This is a CF-language, with grammar

$$\sigma \to y,\ \sigma \to x\sigma,\ \sigma \to x\sigma x. \tag{2}$$

This grammar is ambiguous, for the sentence $x^m y x^n$ has $\binom{m}{n}$ different derivations.

But the ambiguity is a property of the grammar rather than the language; an unambiguous grammar is

$$\sigma \to y,\ \lambda \to y,\ \sigma \to x\lambda,\ \lambda \to x\lambda,\ \sigma \to x\sigma x.$$

In this form a definite order for the application of the rules (2) is laid down.

Example 5. The free monoid X^* has an antiautomorphism which leaves X fixed. This is written $f \mapsto \tilde{f}$ and is called *reversal*; e.g. the reverse of $x^2 yzx$ is $xzyx^2$. Now $\{f\tilde{f} \mid f \in X^*\}$ is a CF-language, but not regular. Taking $X = \{x, y\}$, we have as a generating grammar:

$$\sigma \to x\sigma x,\ \sigma \to y\sigma y,\ \sigma \to 1.$$

By contrast, the language $\{f^2 \mid f \in X^*\}$ is not context-free (it turns out to be context-sensitive).

Example 6. $\{x^n z^n y^n \mid n \geqslant 0\}$ is a CS-language, but not CF. A generating grammar is

$$\sigma \to x\sigma\lambda y,\ \sigma \to xzy,\ y\lambda \to \lambda y,\ z\lambda \to z^2,\ \sigma \to 1.$$

The first two rules generate the words $x^n zy(\lambda y)^{n-1}$, the next moves the λ's past the y's next to z and the fourth rule replaces each such λ by z.

Example 7. The universal language X^* is regular: $\sigma \to \sigma x$ $(x \in X)$, $\sigma \to 1$, and so is the empty language: $\sigma \to \sigma x$.

(iii). We add some remarks on the different types of grammar. The regular languages can be characterized explicitly as follows:

THEOREM 2.1 (Kleene [56]). *The regular languages form the smallest class containing all finite languages and closed under the operations of union, product and* *.

Here the *-operation replaces a language L by

$$L^* = \{u_1 \dots u_n \mid u_i \in L,\ n = 0, 1, 2, \dots\}.$$

All four types of languages are closed under unions, products, reversal, and all except type 2 (CF-languages) are closed under intersection (cf. §4). Further, each type other than 1 (CS-languages) is closed under substitution: If L of type v ($v = 0$, 2, 3) has the alphabet $X = \{x_1, \dots, x_m\}$ and x_1 is replaced by a language of type v, with an alphabet disjoint from X, then the resulting language is again of type v. This follows by performing the substitution on the rewriting rules. E.g. xy and z^* are regular languages; replacing y by z^* we obtain the regular language xz^*. Here we can replace x by y^* and obtain the regular language y^*z^*. On the other hand, we can substitute any language L of type v into x^* and obtain L^* which is again of type v.

Let X, Y be any alphabets and L a language on X. If each $x \in X$ is replaced by a single word in Y, we obtain the image of L under a homomorphism $X^* \to Y^*$. Thus the homomorphic image of any language of type $v \neq 1$ is again of type v. For a CS-language we have closure under substitution, provided that the languages substituted do not contain the empty word. This follows by looking at the derivations (which in a CS-grammar cannot decrease the length).

(iv). Context-sensitive languages are just those languages that can be generated by a grammar with rules of the form $u \to v$, where $l(u) \leqslant l(v)$. For the CS-grammars are of this form, and conversely, given $u \to v$, where $u = u_1 \dots u_r$, $v = v_1 \dots v_s$ and $r \leqslant s$, we take new variables $w_1, \dots, w_r$ and rules

$$u_1 u_2 \dots u_r \to w_1 u_2 \dots u_r \to \dots \to w_1 w_2 \dots w_r \to v_1 w_2 \dots w_r$$
$$\to \dots \to v_1 \dots v_{r-1} w_r \to v_1 \dots v_{r-1} v_r \dots v_s.$$

By treating all rules in this way we obtain a CS-grammar.

It follows from this description that every CS-language is a recursive subset of X^*; therefore $\mathscr{L}_1 \subset \mathscr{L}_0$. However, not every recursive subset of X^* is a CS-language, for we can enumerate the CS-languages: $L_1, L_2, \dots$, and the strings on X: $u_1, u_2, \dots$. Now the set L defined by the rule "$u_i \in L$ if and only if $u_i \notin L_i$, for $i = 1, 2, \dots$" is recursive, but not a CS-language (Chomsky [59]).

The extra power of CS-languages over CF-languages is that rules of the form $uv \to vu$ are admitted in the former; as Chomsky [59] observes, this is really a defect

as far as natural languages are concerned. On the other hand, CF-languages are not enough to describe natural languages. This can be seen by forming sentences of the following type:

Alex came by aeroplane,

Alex and Bill came by aeroplane and boat, respectively,

Alex, Bill and Colin came by aeroplane, boat and car, respectively.

Clearly the series can be continued indefinitely (given sufficient ingenuity), and it corresponds to the sentences

$$u_1v_1,\ u_1u_2v_1v_2,\ u_1u_2u_3v_1v_2v_3, \ldots \tag{3}$$

But as Example 5 above shows, this is not a pattern obtainable by a CF-grammar, which would be limited to generating

$$u_1v_1,\ u_2u_1v_1v_2,\ u_3u_2u_1v_1v_2v_3, \ldots$$

More precisely, any CF-grammar capable of producing the series (3) would also produce many other sentences which one would not want to include in the language.

This type of argument suggests that a natural language such as English is not context-free. On the other hand, many programming languages can be described quite accurately by CF-grammars.

(v). Any CF-language can be generated by a grammar whose rules are all of the form $\alpha \to \beta\gamma$ or $\alpha \to x$ ($\alpha, \beta, \gamma \in V$, $x \in X$). This is called the *Chomsky normal form* (Chomsky [59]); it follows by a double induction (i) on the number of rules not of this form and (ii) the greatest length of any right-hand side. If the grammar contains a rule $\alpha \to a_1 \ldots a_n$ ($a_i \in A$), we replace this by $\alpha \to bc$, $c \to a_2 \ldots a_n$ and $b = a_1$ or $b \to a_1$ according as a_1 is in V or X.

Let G be a grammar of type 0 and $L(G)$ the language generated. Various quite simple questions about $L(G)$ are in general undecidable. E.g., whether a given word w is in $L(G)$ is essentially the halting problem for Turing machines and so is undecidable. For a CS-grammar the corresponding problem is decidable, because $L(G)$ is then recursive, but even here the question whether $L(G)$ is empty is undecidable (it can be reduced to Post's correspondence problem, cf. Gross–Lentin [70]). But for a CF-grammar this problem can be decided:

THEOREM 2.2. *There is an algorithm to determine whether the language generated by a given CF-grammar is empty.*

For if w is a word in $L(G)$ whose derivation takes more than r steps, where r is the number of variables in G, some variable must occur twice on the left of a rule, say there is a step $\alpha \to u$, and later in the derivation there is a step $\alpha \to v$. Now we can get a word with a shorter derivation by substituting the part of the derivation that starts with $\alpha \to v$ into that starting with $\alpha \to u$. Hence if $L(G) \neq \varnothing$, it contains words whose derivation has at most r steps, and we need only check finitely many derivations to see if this is so.

This result can be used to simplify CF-grammars. E.g., given a CF-grammar G, for any variable α we can determine whether there is a derivation starting from α, i.e. whether the grammar G_α obtained by taking α as the sentence symbol σ generates the empty language or not. If $L(G_\alpha) = \varnothing$, we can omit α and all rules involving α from G without affecting the language generated. If $L(G_\alpha)$ is finite, $= \{u_1, \ldots, u_n\}$ say, we can again omit α so long as we add the rules $\sigma \to u_i$ $(i = 1, \ldots, n)$ to G. Thus our grammar is now such that there are infinitely many words derivable from each variable α(unless the whole language is finite). We can also omit a variable α if no derivation starting from σ produces a word containing α. Finally if there is a rule $\alpha \to \beta$, where $\alpha, \beta \in V$, we can again omit it, provided that we add, for each rule $\beta \to u$ $(u \in A^*)$ a rule $\alpha \to u$. We have now reduced our grammar to a form where (i) there are infinitely many derivations starting from each variable, (ii) each variable occurs in some derivation and (iii) no variable occurs by itself on the right of a rule.

A grammar is called *self-embedding* if there is a variable α and a derivation $\alpha \to u\alpha v$, where u, v are non-empty words. If we apply the above reduction to a CF-grammar which is not self-embedding, we obtain a grammar which is again not self-embedding. Our aim will be to show that the language generated by such a grammar is in fact regular. The variables of G can be partially ordered by writing $\alpha < \beta$ whenever there are derivations $\alpha \to u\beta v$ with $u \neq 1$ and with $v \neq 1$ (not necessarily the same derivation). This relation is irreflexive because G is not self-embedding, so we have indeed a partial ordering. Let γ be maximal in the partial ordering, then the rules with γ on the left all have the form

$$\gamma \to \beta u,\ \gamma \to u \qquad (u \in X^*),$$

or all have the form

$$\gamma \to u\beta,\ \gamma \to u \qquad (u \in X^*).$$

In either case the language $L_\gamma = L(G_\gamma)$ is regular. Now replace γ by a new terminal letter z, then we obtain a grammar G' which is again non-self-embedding, but with fewer variables. By induction the language L' it generates is regular. If in L' we substitute L_γ for z we obtain $L(G)$, which is therefore regular. Together with the obvious remark that no regular grammar is self-embedding, this proves

THEOREM 2.3. *A language L is regular if and only if there exists a CF-grammar G generating L which is not self-embedding.*

The same idea is exploited in the following result, sometimes called the pumping lemma (Bar Hillel–Perles–Shamir [**61**], Arbib [**69**]):

THEOREM 2.4. *Let G be a CF-grammar. Then there exist integers p, q such that every word $z \in L(G)$ of length greater than p can be written as $z = xuwvy$, where $uv \neq 1$, $l(uwv) \leqslant q$ and $xu^n wv^n y \in L(G)$ for all $n \geqslant 1$.*

For we can simplify G as before; then the only rules $\alpha \to u$ with $l(u) = 1$ are those where u is a terminal letter. We can omit such a rule, provided that we add all rules

obtained from the existing ones by replacing α by u on the right. The effect on $L(G)$ will be to omit at most one word from the language, namely u. Now let r be the number of non-terminal letters in G, then any derivation with more than r steps has a variable occurring twice on the left. If the longest word obtainable by r derivations has p letters, then any word z longer than p has in its chain of derivations the subchain $\alpha \to u'\alpha v' \to uwv$, where $u' \to u$, $v' \to v$, $\alpha \to w$ are derivations and $uv \neq 1$. Thus z has the form $z = xuwvy$, and by repeating the stage $\alpha \to u'\alpha v'$ of the derivation n times we get $xu^n wv^n y$. Moreover, the derivation $\alpha \to uwv$ can be taken to have at most $r+1$ steps, so uwv has bounded length.

In the case of a regular grammar the terminal letters all occur on the same side of the variables, and the conclusion of Theorem 2.4 simplifies accordingly:

COROLLARY. *Let G be a regular grammar. Then there exists r such that every word w of $L(G)$ of length greater than r has the form $w = xyz$, where $y \neq 1$, $l(yz) \leqslant r$ and $xy^n z \in L(G)$ for all $n \geqslant 1$.*

Here r can again be chosen to be the number of variables in G.

(vi). A great deal has been written on ambiguity. From the definition it is not at all easy to tell whether a given grammar is ambiguous (i.e. whether some word has more than one derivation), in fact the problem is insoluble for general CF-grammars (it can be reduced to Post's correspondence problem, cf. Cantor [**62**], Arbib [**69**]). A CF-language is *unambiguous* if there is an unambiguous CF-grammar generating it, otherwise it is *inherently ambiguous*. An example of the latter is

$$L = \{x^m y^m z^n, x^m y^n z^n \mid m, n \geqslant 0\};$$

this is the union of two CF-grammars and $x^n y^n z^n$ is generated twice, cf. Parikh [**61**], Arbib [**69**].

The following instructive example of an ambiguous CF-grammar is taken from Gross–Lentin [**70**]: The alphabet is $\{e, +, \times\}$ and σ is the only variable. The rules are

$$\sigma \to \sigma+\sigma,\ \sigma \to \sigma\times\sigma,\ \sigma \to e\sigma,\ \sigma \to e.$$

The sentences are just the " calculations " in elementary arithmetic; if we abbreviate e, ee, eee, ... as 1, 2, 3, ..., a typical sentence is $3\times5+4$, and the ambiguity of this grammar is reflected in the fact that this expression has different meanings according to its derivation. The ambiguity can be resolved in several ways; (i) by priority rules ($\times$ precedes $+$) and using brackets: $(3\times5)+4$ or $3\times(5+4)$; or (ii) by writing the operation signs on one side of the variables: $35\times4+$ or $354+\times$.

The second method is often used in abstract algebra (Cohn [**65**], Chapter 3) and it leads to the Łukasiewicz grammar: this has the alphabet $x_0, x_1, \ldots, x_m$, variable σ and rules $\sigma \to \sigma^i x_i$ ($i = 0, 1, \ldots, m$). Here x_μ is to be interpreted as a μ-ary operator. Let us define a homomorphism $X^* \to \mathbf{Z}$ by setting $f(x_i) \to 1-i$, then the words of the language Ł are just the strings u such that $f(u) = 1$ and $f(u') > 0$ for every non-

trivial left factor u' of u. More generally, every string u such that $f(u') > 0$ for each non-trivial left factor u' of u, can be factorized uniquely as the product of $f(u)$ words in Ł (Cohn [65], p. 118).

3. *Machines*

(i). Logical machines form a convenient means of studying recursive functions.. In particular, the Turing machines (Davis [58], Booth [67]) lead precisely to recursively enumerable sets; thus any phrase structure grammar can be obtained by a Turing machine and we would expect the more restricted types 1–3 of grammars to correspond to more special machines. The basic notion is that of a sequential machine:

A *sequential machine* (*Mealy machine*, Hartmanis–Stearns [66], Booth [67]) or also *deterministic automaton* M is given by three non-empty sets (usually finite in what follows): X = *input*, Y = *output* and S = *set of states*, with an *initial state* s_0 say. The action of M is described by two functions, the *transition function* $\delta : S \times X \to S$ (giving the next state) and the *output function* $\lambda : S \times X \to Y$.

M acts *sequentially*, i.e. it reads each word letter by letter, and *synchronously*, i.e. for each $x \in X$ it gives out a $y \in Y$. Generally the input will be a word on X; the output is a word on Y, of the same length, obtained as follows. Define mappings δ', λ' by

$$\delta'(s, 1) = s, \delta'(s, ux) = \delta(\delta'(s, u), x) \quad s \in S, \quad x \in X, \quad u \in X^*,$$

$$\lambda'(s, 1) = 1, \lambda'(s, ux) = \lambda'(s, u)\,\lambda(\delta'(s, u), x).$$

This defines a mapping $\delta' : S \times X^* \to S$ which extends δ and defines a representation of the free monoid X^* by mappings of S. Likewise $\lambda' : S \times X^* \to Y^*$ is a mapping extending λ. We may therefore without risk of confusion omit the dashes from δ', λ'.

To operate the machine we start with an input $u \in X^*$ and with the machine in the initial state s_0. The machine then moves into the state $\delta(s_0, u)$ and produces the output $\lambda(s_0, u)$. We see that an automaton differs from a Turing machine chiefly in having no specific memory (tape). If S, λ are absent from M and instead we have a set $F \subseteq S$ of *final states*, we speak of an *acceptor*. Now there is no output, but merely a set of final states; the set of words in X which take us from s_0 to a state in F is the set *accepted* by M.

Algebraically an acceptor may be described as a set S with a representation of the free monoid X^* by mappings of S into itself. An acceptor may also be represented as a graph: we take the set of states as nodes, with a segment $s_i \to s_j$ for each letter x such that $\delta(s_i, x) = s_j$. Similarly an automaton may be represented by a graph, with a segment $s_i \to s_j$ labelled by x/y if $\delta(s_i, x) = s_j$ and $\lambda(s_i, x) = y$. The importance of acceptors for us rests in

THEOREM 3.1. *A language is regular if and only if it is the precise set accepted by some (finite) acceptor.*

This follows easily using Theorem 2.1 (Arbib [69]). As an example, here is an acceptor for the language $\{xy^n \mid n \geqslant 0\}$ (Example 1). There are three states: s_0 (initial), s_1, s_2 and only s_1 is final. The transition function δ is given by the table

δ	x	y
s_0	s_1	s_2
s_1	s_2	s_1
s_2	s_2	s_2

Let M, M' be two machines with inputs X, X', outputs Y, Y' and sets of states S, S'. A *homomorphism* $f: M \to M'$ is a triple of mappings $X \to X'$, $Y \to Y'$, $S \to S'$ all denoted by f for short, such that the following diagrams commute:

$$\begin{array}{ccc} S \times X & \xrightarrow{\delta_M} & S \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f} \\ S' \times X' & \xrightarrow[\delta_{M'}]{} & S' \end{array} \qquad \begin{array}{ccc} S \times X & \xrightarrow{\lambda_M} & Y \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f} \\ S' \times X' & \xrightarrow[\lambda_{M'}]{} & Y' \end{array}$$

In the special case when $X = X'$, $Y = Y'$ and f is the identity of X and Y, f is called a *state homomorphism.*

Given any machine, we can often find a simpler machine doing the same work by taking a state homomorphism to a machine with fewer states. If the number of states cannot be diminished in this way, the machine is said to be *reduced.* Every machine has a reduced homomorphic image, which can be constructed by identifying states having the same effect.

Instead of using the functions δ, λ to describe M we can also take the set of all quadruples (s_i, x, y, s_j) such that $\delta(s_i, x) = s_j$ and $\lambda(s_i, x) = y$. In this form it is evident how to define the *adjoint* M^* of M: it has input Y, output X, the same set of states S as M and quadruples (s_j, y, x, s_i) corresponding to each (s_i, x, y, s_j) for M. The functions δ, λ defining M^* will in general be many-valued, i.e. M^* will usually be a *non-deterministic* automaton; however, it is deterministic whenever M was reduced (Arbib [69]). Using adjoints it is easy to show that the reverse of any regular language is again regular.

(ii). Acceptors can be used to estimate the number of words in a language. To see this we note the following description of automata by matrices.

Let M be an automaton with N states $s_1, \ldots, s_N$,then M can be described by a set of $N \times N$ matrices $P(x \mid y)$ $(x \in X, y \in Y)$, where

$$P(x \mid y) = (p_{ij}(x \mid y)),\ p_{ij}(x \mid y) = \begin{cases} 1 & \text{if } \delta(s_i, x) = s_j \text{ and } \lambda(s_i, x) = y, \\ 0 & \text{otherwise.} \end{cases}$$

The $P(x \mid y)$, called the *transition matrices,* describe the action of M completely.

More generally, we can define $P(u \mid v)$ recursively for any $u \in X^*$, $v \in Y^*$ by the equations

$$P(ux \mid vy) = P(u \mid v)\, P(x \mid y), \quad P(u \mid v) = 0 \quad \text{if} \quad l(u) \neq l(v).$$

It is easily verified that for any $u, u' \in X^*$, $v, v' \in Y^*$ we have

$$P(uu' \mid vv') = P(u \mid v)\, P(u' \mid v'). \tag{1}$$

For each $x \in X$, let us write

$$P(x) = \sum_y P(x \mid y),$$

where y ranges over Y, then $P(x)$ is a matrix with precisely one 1 in each row and the rest 0's. In particular, for an acceptor we have only the matrices $P(x)$, and (1) takes the form

$$P(uu') = P(u)\, P(u').$$

Thus P is the matrix representation of X^* acting on S. Let π be the row vector describing the initial state: $\pi_i = 1$ if s_i is the initial state and $\pi_i = 0$ otherwise, and let f be the column vector with entry 1 for final and 0 for non-final states, then for any $u \in X^*$

$$\pi P(u) f = \begin{cases} 1 & \text{if } u \text{ is accepted,} \\ 0 & \text{if not.} \end{cases}$$

Put $A = \sum_x P(x)$, then $a_{ij}(n)$, the (i, j)-entry of A^n, represents the number of words of length n which send s_i to s_j. In terms of the graph of M it is the number of paths of length n from s_i to s_j; in particular, $a_{ii}(n)$ is the number of loops (closed circuits) of length n based at s_i. Let us write $\lambda(n)$ for the number of words of length n in the language, and write

$$f(t) = \sum_0^\infty \lambda(n)\, t^n$$

for the length generating function, then

$$f(t) = \sum \pi A^n t^n f = \pi(I - At)^{-1} f.$$

To get an asymptotic value for $\lambda(n)$ we take the characteristic equation of A in the form

$$A^N + c_1 A^{N-1} + \ldots + c_N = 0,$$

then for $n \geqslant N$,

$$a_{ij}(n+1) + c_1 a_{ij}(n) + \ldots + c_N a_{ij}(n-N+1) = 0.$$

Taking s_1 to be the initial state, we have $\lambda(n) = \sum a_{1j}(n)$, where the sum is over all j such that s_j is a final state; hence we obtain the following recursions relation for the λ's:

$$\lambda(n+1) + c_1 \lambda(n) + \ldots + c_N \lambda(n-N+1) = 0.$$

The characteristic values are just the eigenvalues of A: $\lambda_1, \ldots, \lambda_N$ say. Assuming them

to have distinct moduli for the moment, we obtain as the general solution

$$\lambda(n) = b_1 \lambda_1{}^n + \ldots + b_N \lambda_N{}^n, \tag{2}$$

where the b_i are constants. If the λ_i are so numbered that $|\lambda_1| > |\lambda_i|$ for all i, then $\lambda(n)/\lambda_1{}^n \to b_1$ as $n \to \infty$, and

$$\lim \frac{\log \lambda(n)}{n} = \log |\lambda_1|. \tag{3}$$

In general λ_1 is an r-fold eigenvalue, and b_1 in (2) is now a polynomial of degree r in n. Thus $\log \lambda(n) \sim n.\log|\lambda_1| + r.\log n + c$, and (3) still holds.

The limit on the left of (3) is known as the *channel capacity* in information theory; for this reason $\log|\lambda_1|$ may be regarded as the information capacity of the language. If the logarithms are taken to base 2, this gives the capacity of the language in bits per letter. As an example, suppose that the graph of the acceptor has just one loop, of length r say. The number of loops of length n is $tr\, A^n$, hence $\sum \lambda_i{}^\nu = \delta_{\nu r}$ and it follows that A has the characteristic equation $A^N - A^{N-r} = 0$, and $\log|\lambda_1| = 0$.

There is a curious property of regular languages. Let m be the number of letters in the alphabet X, then the universal language X^* has $\mu(n) = m^n$ words of length n, so $\lambda_1 = m$ in this case. Given any regular language L, its complement L' is again regular, and if $\lambda(n)$, $\lambda'(n)$ are the number of words of length n in L, L' respectively, then

$$\lambda(n) + \lambda'(n) = m^n. \tag{4}$$

Hence

$$\frac{\lambda(n)}{\mu(n)} \sim c.(\lambda_1/m)^n,$$

where c is at most a polynomial in n. Now if $|\lambda_1| < m$, then $c.(\lambda_1/m)^n \to 0$ as $n \to \infty$, hence by (4), $\lambda'(n)/\mu(n) \to 1$. This means that L' has a greatest eigenvalue of absolute value m. We obtain

THEOREM 3.2. *Let L be a regular language on m letters; either L has capacity* log *m, or the probability that a sufficiently long string of letters forms a word in L is arbitrarily close to zero.*

(iii). We now turn to context-free languages. If we take a typical CF-language such as $\{x^n y^n \mid n \geqslant 0\}$ and try to build an acceptor for it we find that we are hampered by the fact that the acceptor has no memory. Here we see again that English is not a regular language, since we clearly need a memory to parse (or form) such pairs of sentences as

$$\begin{matrix}\text{Those} \\ \text{He}\end{matrix} \text{ whom the gods love } \begin{matrix}\text{die} \\ \text{dies}\end{matrix} \text{ young.}$$

The memory to be described is of a rather simple sort, a " first-in, last-out " store, where we only have access to the last item in the store.

Formally, a *pushdown acceptor* (PDA for short) is an acceptor which in addition to its set S of states and input alphabet X has a set Σ of *store symbols*, with initial symbol λ_0, and a transition function δ on $S \times X \times \Sigma$ with values in $S \times \Sigma^*$, but for a given triple of arguments there may be several or no values (this machine is non-deterministic). At any stage the machine is described by a triple (s_i, w, α), where $s_i \in S$, $w \in X^*$, $\alpha \in \Sigma^*$. We apply δ to the triple consisting of s_i, the first letter of w and the last letter of α. If $w = xw'$, $\alpha = \alpha'\lambda$ say, and (s_j, β) is a value of $\delta(s_i, x, \lambda)$, then

$$(s_i, xw', \alpha'\lambda) \to (s_j, w', \alpha'\beta)$$

is a possible move. We say that $w \in X^*$ is *accepted* by the machine if starting from (s_0, w, λ_0) there is a series of moves to take us to $(s_r, 1, \gamma)$, where s_r is a final state. Now we have

THEOREM 3.3. *The context-free languages constitute the precise class of sets accepted by pushdown acceptors.*

For a proof we refer to Arbib [**69**]. As an example we describe a PDA for $\{x^n y^n \mid n \geqslant 0\}$. Its states are s_0 (initial state), s_1, s_2 where s_2 is final. The store symbols are λ (initial symbol), μ, ν. We give the values for $(s_i, ., .)$ in the form of a table for each s_i:

s_0	λ	μ	ν
x	$s_0\mu$	$s_0\mu\nu$	$s_0\nu^2$
y		$s_2\mu$	s_1

s_1	λ	μ	ν
x			
y		$s_2\mu$	s_1

Blanks and the remaining values (for s_2) are undefined. It is easily seen that the strings accepted are just $x^n y^n$; note how the store acts as a memory, remembering how many factors x have been taken off. If we think of the store arranged vertically, at each stage we remove the topmost symbol and add a number of symbols to the top, just like a stack of plates in a cafeteria; this explains the name.

Another example of a CF-language is the set L_1 of all strings u in x, y such that in any left-hand factor of u the degree in x is at least equal to the degree in y. It is easy to construct a PDA for L_1 and the reader may like to try this for himself. A somewhat harder example is the set L_2 of strings in x, y whose degree in x is at least equal to the degree in y. For L_2 the probability that a string of given even length is in L_2 is 1/2. By Theorem 2.4 Corollary, L_2 is not regular; likewise L_1 is not regular.

(iv). We can think of a PDA as a Turing machine, by regarding the store as part of the tape†. An estimate of the length of tape needed may be given as follows. Let n be the length of the input word, a the greatest length of any word added to the

† A PDA is most naturally regarded as a 2-tape Turing machine, but of course this is equivalent to the usual (1-tape) sort.

store at any stage and b the length of tape taken up with instructions, then the total length of tape needed is $an+b$, a linear function of n. We shall find that a more general type of acceptor, with a similar limitation on the length of tapes, can be used for general CS-languages.

A *linear-bounded automaton* is an automaton in which the input and output alphabet are the same, X say, with an initial state s_0 and final states, and with a tape on which the input word is written. The configuration of the machine at any stage is described by a state s_i and a square of the tape being scanned, and in addition to the transition function δ and output function λ there is a transport function

$$\tau : S \times X \to \{L, R, O\},$$

where L, R, O denotes motion of the tape by one square to the left, right or not at all. As input we take the symbol on the square being scanned and replace it by the output. In general these functions will again be many-valued partial functions. A word w is *accepted* if there is at least one computation starting with w printed on the tape, the leftmost letter being scanned, and in the initial state s_0, and terminating with the last letter being read as the machine enters a final state. Such a machine differs from the general Turing machine in having a tape length which is bounded by a linear function of the input word.

THEOREM 3.4. (Landweber [63], Kuroda [64]). *The context-sensitive languages constitute the precise class of languages accepted by linear-bounded automata.*

E.g. it is not hard to construct a linear-bounded automaton to accept

$$\{x^n y^n z^n \mid n \geqslant 0\}.$$

Intuitively we first move across the word to check that it is a block of x's followed by a block of y's followed by a block of z's. Now erase (i.e. replace by a further, neutral symbol) one z, one y, one x and recommence. The formal description is rather longer and will be omitted.

(v). The range of languages can be extended in another direction by using stochastic automata. We recall that any automaton can be described by a set of transition matrices $P(x \mid y)$; here the $P(x \mid y)$ are non-negative (i.e. with non-negative entries) and such that $P(x) = \sum_y P(x \mid y)$ has one entry 1 in each row and the rest 0's. If, more generally we take $P(x \mid y)$ to be a non-negative matrix such that $P(x)$ is *stochastic* i.e. has row sums 1, we have a *stochastic automaton* or *acceptor* as the case may be.

In a stochastic acceptor A it is natural to replace the initial state by an *initial distribution* π, i.e. a row vector whose components are non-negative and add up to 1; by f we denote again the column vector with 1 for final and 0 for non-final states. With each word $u \in X^*$ we can associate a real number between 0 and 1:

$$\lambda_u = \pi P(u) f,$$

representing the " strength of acceptance " of the word u by A. We say that u is *λ-accepted* by A if $\pi P(u) f > \lambda$.

Given $0 \leqslant \lambda \leqslant 1$ and $L \subseteq X^*$, we say that L is *λ-stochastic* if there is a stochastic acceptor A such that L is the language λ-accepted by A, i.e.

$$L = L(A, \lambda) = \{u \in X^* \mid \pi P(u) f > \lambda\}.$$

Now a *stochastic language* is a language that is λ-stochastic for some λ. Every 0-stochastic language or more generally, every λ-stochastic language with a deterministic acceptor is regular; but in general a stochastic language need not even be of type 0.

As an example of a stochastic acceptor let us take the alphabet $X = \{0, 1, \ldots, m-1\}$, where $m \geqslant 2$, and define an acceptor B with two states s_1 (initial state), s_2 (final state) and

$$P(x) = \begin{pmatrix} 1-x/m & x/m \\ 1-(x+1)/m & (x+1)/m \end{pmatrix} \quad x = 0, 1, \ldots, m-1.$$

Each word $u = x_1 \ldots x_r$ may be represented in m-adic form as $0.x_r x_{r-1} \ldots x_1$. Then the language λ-accepted by B is

$$L(B, \lambda) = \{x_1 \ldots x_r \in X^* \mid 0.x_r \ldots x_1 > \lambda\}.$$

It is not hard to verify that $L(B, \lambda)$ is regular if and only if λ is rational (Paz **[66]**, Claus **[71]**), so that we obtain languages that are not regular by choosing λ irrational. Moreover, since $L(B, \lambda) \neq L(B, \lambda')$ for $\lambda \neq \lambda'$, we have in fact uncountably many stochastic languages. In particular this shows that not every stochastic language is of type 0. An explicit example of a stochastic language that is not regular is

$$\{x^m y^n \mid n-2 \geqslant m \geqslant 0\} \text{ (cf. Claus [71])}.$$

Stochastic automata are of interest because real machines are never fully determinate, but in fact show stochastic behaviour. Moreover, if we drop " stochastic ", i.e. allow arbitrary real matrices $P(x)$ in the above definition, the remarkable fact emerges that the languages so obtained are nevertheless stochastic. This result is due to Turakainen **[69]** (cf. also Claus **[71]**).

4. *Transductions*

(i). To formalize the notion of translation from one language to another one introduces transductions. A *transduction* from X^* to Y^* is just a subset T of $X^* \times Y^*$. Given a transduction T from X^* to Y^* we can apply T to any language L on X by forming

$$T(L) = \{v \in Y^* \mid (u, v) \in T \text{ for some } u \in L\}.$$

This is called the *image* of L under T. It is clear how transductions $T \subseteq X^* \times Y^*$ and $U \subseteq Y^* \times Z^*$ can be composed (in the usual manner of relations, cf. e.g. Cohn **[65]**) to give a transduction $TU \subseteq X^* \times Z^*$, and the inverse $T^{-1} \subseteq Y^* \times X^*$ can be formed.

A transduction T is said to be *regular* or a *K-transduction* if there is an automaton A with a pair of states s_1, s_2 (not necessarily distinct) such that T consists precisely of those pairs (u, v) for which the input u in state s_1 leads to output v in state s_2; A is also called a *transducer* for T. Thus an automaton with N states defines N^2 transductions;

a K-transduction by a machine with a single state is just a homomorphism $X^* \to Y^*$. It is clear that any product of regular transductions is again regular.

What we have just defined should more precisely be called a *right regular* transduction, because the action of X^* on A is a right action. Left regular transductions may be defined similarly. To be explicit, let $\sim$ be the reversal on

$$X^* \times Y^* : (u, v) \mapsto (\tilde{u}, \tilde{v}),$$

then R is left regular if and only if $\tilde{R}$ is right regular. It can be shown that every regular language in Y^* can be obtained from X^* be applying first a right and then a left regular transduction (cf. Schützenberger [**61**]).

(ii). Let us consider the effect of a transduction T on a CF-language L. The operation of T may be described in terms of quadruples (s_i, x, y, s_j), as we have seen; we shall again denote the initial state by s_0. The rules of L can be written in Chomsky normal form as $\alpha \to \beta\gamma$ or $\alpha \to x$ ($\alpha, \beta, \gamma \in V$, $x \in X$). The image L' of L has again a CF-grammar, with non-terminal variables of the form (s_i, x, s_j) and (s_i, α, s_j) and rules

(i) $\sigma \to (s_0, \sigma, s_i)$ for all i such that s_i is final,

(ii) if $\alpha \to \beta\gamma$ in L, then $(s_i, \alpha, s_j) \to (s_i, \beta, s_k)(s_k, \gamma, s_j)$ for all i, j, k,

(iii) if $\alpha \to x$ in L, then $(s_i, \alpha, s_j) \to (s_i, x, s_j)$ for all i, j,

(iv) if (s_i, x, y, s_j) is in T, then $(s_i, x, s_j) \to y$.

It may be verified that $T^{-1}(L') = L \cap \Lambda$, where Λ is the regular language accepted by T. Thus T accepts the intersection of L with the maximal regular language it accepts.

Now given any regular language Λ, there is an acceptor for Λ, which can be turned into an automaton by recopying the input word. Thus T is a subset of the diagonal and $L' = L \cap \Lambda$. This proves

THEOREM 4.1 (Bar Hillel–Perles–Shamir [**61**]). *The intersection of a CF-language and a regular language is a CF-language.*

In the same way one can show that the image of a regular language by a transducer is regular and hence the intersection of two regular languages is regular. By contrast the intersection of two CF-languages need not be CF, e.g.

$$\{x^m y^n z^n \mid m, n \geqslant 0\} \cap \{x^m y^m z^n \mid m, n \geqslant 0\} = \{x^n y^n z^n \mid n \geqslant 0\}.$$

5. *Monoids*†

(i). Let M be a monoid and S a finite set with a right *M-action*, i.e. to each pair $(s, x) \in S \times M$ there corresponds an element $sx \in S$ such that $s(xy) = (sx)y$ and $s1 = s$. We fix $s \in S$ and define a relation $\mathfrak{r}_s$ on M as follows:

$$x \,\mathfrak{r}_s\, y \quad \text{if and only if} \quad sx = sy.$$

† All monoids in this section are understood to be finitely generated.

Clearly this is an equivalence on M, which moreover is *right admissible*, i.e.

$$x\,\mathfrak{r}_s\,y \Rightarrow xz\,\mathfrak{r}_s\,yz \quad \text{for all} \quad x, y, z \in M.$$

The classes of this equivalence correspond just to the images of s under the action of M and, since S is finite, the *index* of $\mathfrak{r}_s$, i.e. the number of $\mathfrak{r}_s$-classes, is also finite.

If we do this for each $s \in S$ and put

$$\mathfrak{q} = \bigcap \mathfrak{r}_s, \tag{1}$$

we obtain another equivalence which is left as well as right admissible: $x\,\mathfrak{q}\,y$ means that $sx = sy$ for all $s \in S$, hence $szx = szy$ for all $z \in M$. Thus $\mathfrak{q}$ is a *congruence* on M and we can form the quotient monoid $M/\mathfrak{q}$. To obtain an interpretation for $\mathfrak{q}$, let us write $Map(S)$ for the monoid of all mappings of the set S into itself, then the right M-action on S defines a monoid homomorphism

$$\phi : M \to Map(S),$$

and now the congruence $\mathfrak{q}$ defined in (1) is just the kernel of ϕ, while $M/\mathfrak{q}$ is the image (up to isomorphism). Since $Map(S)$ is finite, this shows that $\mathfrak{q}$ has again finite index.

We saw that each $\mathfrak{r}_s$ is a right admissible equivalence of finite index. Conversely, let $\mathfrak{r}$ be any right admissible equivalence on M of finite index and write $S = M/\mathfrak{r}$ for the set of $\mathfrak{r}$-classes. By definition of $\mathfrak{r}$, right multiplication by $x \in M$ maps each $\mathfrak{r}$-class again into an $\mathfrak{r}$-class, thus we have a right M-action on S, and this shows that the M-actions on finite sets correspond to right admissible equivalences on M of finite index.

(ii). These remarks can be applied to regular languages, bearing in mind that the acceptors for these languages are just finite sets with an X^*-action. Thus let A be an acceptor with initial state 0 and set of final states F, and let L be the language accepted by A. Then L is the union of all the $\mathfrak{r}_0$-classes mapping 0 to some state in F; hence L is the union of certain $\mathfrak{r}_0$-classes, where $\mathfrak{r}_0$ is a right admissible equivalence of finite index. Conversely, if L is the union of a finite number of classes of a right admissible equivalence of finite index, we can form an acceptor for the language by taking the quotient with respect to this equivalence. In this way we obtain another criterion for the regularity of a language:

Given $L \subseteq X^*$, we form the least right admissible equivalence $\mathfrak{r}(L)$ on X^* for which L is a union of $\mathfrak{r}(L)$-classes; $\mathfrak{r}(L)$ may be defined explicitly by the rule

$$u\,\mathfrak{r}(L)\,v \quad \text{if and only if} \quad uw \in L \Leftrightarrow vw \in L \quad \text{for all} \quad w \in X^*.$$

It is called the *Nerode equivalence* associated with L on X^*. Now it is clear that we have (Nerode [58], Teissier [51])

THEOREM 5.1. *A language L in X is regular if and only if the Nerode equivalence $\mathfrak{r}(L)$ on X^* is of finite index.*

A similar discussion can be carried through for the least congruence $\mathfrak{q}$ such that L is a union of $\mathfrak{q}$-classes. This is the *Myhill congruence* $\mathfrak{q}(L)$ associated with L and

may be defined by the rule

$$u\,\mathfrak{q}(L)\,v \quad \text{if and only if } yuz \in L \Leftrightarrow yvz \in L \quad \text{for all} \quad y, z \in X^*.$$

As before we obtain (Myhill [57], Teissier [51]):

THEOREM 5.2. *A language L is regular if and only if the Myhill congruence $\mathfrak{q}(L)$ on X^* is of finite index.*

A slight reformulation leads to the

COROLLARY. *A language L is regular if and only if there is a homomorphism $f: X^* \to M$ to a finite monoid M such that $L = f^{-1}(f(L))$.*

E.g. the regular languages on one letter are all of the form $x^{an+b} (n = 0, 1, 2, \ldots)$, for fixed a, b. The language $\{x^m y^n \mid n \geqslant m \geqslant 0\}$ is easily seen not to be regular, for if $i < j$, we have $x^i y^{j-1} \in L$ but $x^i y^{i-1} \notin L$, hence x^i, x^j lie in different classes of the Nerode equivalence, which therefore has infinite index.

(iii). For some purposes (e.g. transductions) it is useful to consider direct products of free monoids; regularity corresponds here to several inequivalent notions.

Given any monoid M, a subset L of M is said to be *rational* if it can be obtained from finite subsets of M by the operations of union, product and * (i.e. forming from P the submonoid P^* generated by P). When $M = X^* \times Y^*$, this can be shown to agree with the notion of a regular transduction defined in §4.

A subset L of a monoid M is said to be *recognizable* if it is the union of congruence classes of a congruence of finite index. Let us call a topological monoid *coarse* if the topology is induced by a homomorphism to a finite monoid (with the discrete topology) then a recognizable subset of M is a subset which is closed in some coarse topology on M.

It is easily seen that if $f: M \to N$ is a homomorphism of monoids, then f preserves rational sets, while f^{-1} preserves recognizable sets. Moreover, by Theorems 2.1 and 5.2, the notions "rational" and "recognizable" coincide for free monoids. It follows that any recognizable set is rational. For if L is recognizable in M, and $f: F \to M$ is a surjective homomorphism from a free monoid F (which always exists), then $f^{-1}(L)$ is recognizable in F, hence rational, and so $L = f(f^{-1}(L))$ is rational. But in general these notions are distinct, e.g. on a direct product of free monoids we have the following criterion (cf. Fliess [70] where this result is ascribed to Mezei):

THEOREM 5.3. *A subset of $M_1 \times M_2$ is recognizable if and only if it is a finite union of products $R_1 \times R_2$ of recognizable subsets of M_1 and M_2.*

This follows because any coarse topologies on M_1 and M_2 (always compatible with the monoid structure) lead to a coarse topology on $M_1 \times M_2$, via the projections $M_1 \times M_2 \to M_i$ and conversely, via the injections $M_i \to M_1 \times M_2$.

(iv). Each CF-language may be written as a monoid equation, by replacing the non-terminal variable α by L_α, the set of words derivable from α. Thus each non-terminal variable is replaced by a subset of X^* and σ becomes the language itself. It will be convenient to use $+$ as union sign and to identify 1-element sets with their members. Let us take e.g. $\{x^m y x^n \mid m \geqslant n \geqslant 0\}$ (Example 4); replacing σ, λ by L, A respectively, we have

$$L = y + xA + xLx, \quad A = y + xA.$$

We shall apply this description to a particularly simple type of regular language. Let L be given by a grammar with alphabet $X = \{x_1, \ldots, x_m\}$, non-terminal variables $\sigma, \alpha_1, \ldots, \alpha_m$ and rules such that σ occurs in just one rule on the left, while the right-hand side of any rule is of the form x_i or $x_i \alpha_i$. In equational form (writing A_i for α_i) we have

$$L = x_{i_1} A_{i_1} + \ldots + x_{i_r} A_{i_r} + x_{k_1} + \ldots + x_{k_t},$$

$$A_i = \delta_{ij_1} x_{j_1} + \ldots + \delta_{ij_s} x_{j_s} + \delta_{i1} x_1 A_1 + \ldots + \delta_{im} x_m A_m.$$

Here $\delta_{ij} = 1$ or 0 according as A_i contains a word beginning with x_j or not,

$$I = \{x_{i_1}, \ldots, x_{i_r}\}$$

are the initial letters of words in L, $J = \{x_{j_1}, \ldots, x_{j_s}\}$ are the final letters of words in L and $I \cap J = \{x_{k_1}, \ldots, x_{k_t}\}$. Thus L may also be described as

$$L = \{IX^* \cap X^* J\} \setminus X^* D X^*,$$

where $D \subseteq (X^*)^2$ is the set of pairs $x_i x_j$ such that $\delta_{ij} = 0$. A language of this form is called a *standard K-language* or also a *local language*; to test if a word belongs to it we need not look at the word as a whole but only at the first and last letter and at pairs of adjacent letters. Thus if we form a graph on X as set of nodes, with a segment from x to y whenever $xy \notin D$, then the words of the language are represented by all the paths beginning in I and ending in J.

The importance of K-languages stems from the following result (Myhill **[57]**):

THEOREM 5.4. *Given any regular language L, there exists a standard K-language L_0 and a homomorphism ϕ such that $\phi(L_0) = L$.*

The proof proceeds by taking a regular grammar for L and associating with each letter x_i occurring on the right of a rule the symbol (x_i, σ), with each pair $x_i \alpha_j$ the symbol (x_i, α_j) $[x_i, \alpha_j]$ and with σ the symbol $[\sigma]$. We then obtain a standard K-language with alphabet (x_i, α_j) and non-terminal variables $[x_i, \alpha_j]$, $[\sigma]$, while $(x_i, \alpha_j) \mapsto x_i$ is the required homomorphism.

There is an analogous characterization of regular transductions (Nivat **[68]**):

THEOREM 5.5. *Any homomorphism of a standard K-language into $X^* \times Y^*$ is a regular transduction, and conversely, every regular transduction may be obtained in this way, as homomorphic image of a standard K-language.*

This result shows that the inverse of a regular transduction is again regular, a fact which was not obvious from the definition.

(v). There is a similar analysis of CF-languages, with the help of an important special class, the Dyck languages. We begin with some general remarks on homomorphisms of monoids.

Given any homomorphism of monoids, $f: M \to N$, the inverse image of 1 under f is called the *kernel* of f, $\ker f$. This kernel is always a submonoid, and if M is free then so is $\ker f$. To see this we recall the criterion (Cohn [**62, 71**]):

A submonoid T of a free monoid M is free if and only if for any $a, b, a', b' \in T$ satisfying $ab' = ba'$ there exists $c \in T$ such that $a = bc$ or $b = ac$.

Given $f: M \to N$, where M is free, and $ab' = ba'$, where $a, b, a', b' \in \ker f$, we have $a = bc$ or $b = ac$, say the former, for some $c \in M$, because M is free. Now $1 = f(a) = f(b)f(c) = f(c)$, hence $c \in \ker f$.

We shall take M to be free on $X \cup X'$, where X' is a set bijective with X by the rule $x \leftrightarrow x'$, and take $f: M \to F_X$ to be the homomorphism from M to F_X, the free group on X, in which $x \mapsto x$ and $x' \mapsto x^{-1}$ $(x \in X)$. The kernel is written D^* and is called the *Dyck language* on $X \cup X'$. By what has been said, D^* is a free monoid; its elements are all words in $X \cup X'$ which reduce to 1 on omitting all occurrences of xx' and $x'x$. The free generating set of D^* is the set D of words of the form xfx' or $x'fx$, where $f \in D^*$. This justifies the notation D^*.

We obtain a submonoid D_r of D^* by allowing only those words which reduce to 1 on omitting occurrences of xx'; this is called a *restricted Dyck language*. It may be obtained as the kernel of a homomorphism to a certain monoid (the bicyclic monoid and its polycyclic generalizations, cf. Clifford–Preston [**61**] and Nivat [**68**]), and so is again free. An example of a restricted Dyck language is the set of bracket formations obtainable from a given set of pairs of brackets $(_i,)_i$ (cf. Cohn [**65**], p. 268).

The Dyck language is context-free, a CF-grammar being

$$\sigma \to \sigma\sigma,\ \sigma \to x\sigma x',\ \sigma \to x'\sigma x\ (x \in X),\ \sigma \to 1.$$

This grammar turns out to be ambiguous, but an unambiguous grammar is easily constructed (Gross-Lentin [**70**], p. 187).

Let C be a standard K-language, then the intersection $C \cap D^*$ with a Dyck language is necessarily a CF-language, as we saw in §4. Any language of the form $C \cap D^*$, where C is a standard K-language and D^* is a Dyck language, possibly restricted, is called a *standard CF-language*. Thus a standard CF-language is subject to two kinds of restriction, those due to C, which are "local", while those due to D^*, though more subtle, are the same in all cases. What gives standard CF-languages their importance is the following result (Chomsky–Schützenberger [**63**]):

THEOREM 5.6. *Every CF-language L can be obtained as a homomorphic image of a standard CF-language. Thus there is a Dyck language D^* (possibly restricted) and a standard K-language C, with a homomorphism ϕ such that $L = \phi(C \cap D^*)$.*

The proof consists essentially in enclosing all occurrences of variables on the right of the rewriting rules in brackets, serially numbered: $(_1,)_1$, $(_2,)_2$, ..., and replacing each terminal letter x by xx'. We thus obtain a Dyck language which is also regular (in fact it is a standard K-language): each word is bracketed so that it can be tested by a finite acceptor. Now the homomorphism consists in setting the added symbols equal to 1. (cf. Arbib [**69**] for a detailed proof). E.g., to obtain the mirror language $L = \{f\tilde{f} \mid f \in X^*\}$, let C be the standard K-language on $X \cup X'$ with initials $x \in X$ and forbidden pairs $x'y$ $(x, y \in X)$, then $C \cap D^* = \{f\tilde{f} \mid f \in X^*\}$, where $f \mapsto \tilde{f}$ is the anti-isomorphism $X \to X'$ defined by $\bar{x} = x'$. Now the homomorphism $x \mapsto x$, $x' \mapsto x$ maps $C \cap D^*$ to L.

This theorem throws some light on the structure of CF-languages: while the class of standard CF-languages is closed under intersections, the class of all CF-languages is not, so the irregularities are in some sense due to the homomorphism. This is borne out by the fact that, given two homomorphisms $f, g : X^* \to Y^*$ between free monoids, there is no general procedure for determining whether a non-empty word u in X exists such that $f(u) = g(u)$; in essence this is again Post's correspondence problem (Gross–Lentin [**70**]).

6. *Power series*

(i). The description of languages by formal power series in non-commuting indeterminates, in many ways the simplest and most natural, is due to Schützenberger [**61**]. Starting from any commutative ring K, we can form the free K-algebra on X, $K\langle X\rangle$ as the monoid algebra of X^* over K. Its power series completion is denoted by $K\langle\langle X\rangle\rangle$; since X is finite, no ambiguity arises (for infinite X the structure of $K\langle\langle X\rangle\rangle$ depends on whether the degrees of the elements of X are bounded).

An element of $K\langle\langle X\rangle\rangle$ is said to be *rational* if it can be obtained from the elements of $K\langle X\rangle$ by repeated division (by elements with constant term 1) together with the other ring operations. As Schützenberger [**62**] has observed, this is so if and only if the element can be obtained as a component of the solution of a matrix equation

$$Au = a, \tag{1}$$

where a is a column vector in $K\langle X\rangle$ and A a matrix which is invertible over $K\langle\langle X\rangle\rangle$. This observation, suitably reformulated, turns out to be valid for entirely arbitrary rings, and it lies at the basis of the author's study of homomorphisms of rings into skew fields (Cohn [**71**], Chapter 7).

Let $\varepsilon : K\langle\langle X\rangle\rangle \to K$ be the augmentation mapping, i.e. the K-algebra homomorphism defined by $\varepsilon(X) = 0$. We observe that a square matrix A over $K\langle X\rangle$ is invertible over $K\langle\langle X\rangle\rangle$ precisely when its image $\varepsilon(A)$ is invertible over K. In that case we can multiply by $\varepsilon(A)^{-1}$ to make a reduction to the case where $\varepsilon(A) = I$. Thus we may in (1) take $A = I - B$, where B has entries with zero constant terms; (1) now becomes

$$u = Bu + b, \tag{2}$$

or in components: $u_i = \sum b_{ij} u_j + b_j$, where $b_j, b_{ij} \in K\langle X\rangle$, $\varepsilon(b_{ij}) = 0$. The rational

power series form a subalgebra of $K\langle\langle X\rangle\rangle$, denoted by $K_{\mathrm{rat}}\langle X\rangle$.

An element $f \in K\langle\langle X\rangle\rangle$ is said to be *algebraic*, if it is of the form $f = u_1 + \gamma$, where $\gamma \in K$ and u_1 is a component of the solution of a system of equations

$$u_i = \phi_i(u, x), \tag{3}$$

where ϕ_i is a (non-commutative!) polynomial in the u's and x's without constant term or linear term in the u's. It is easily seen that we can solve for u_i by successive substitution and obtain a unique solution in $K\langle\langle X\rangle\rangle$. The algebraic power series form again a subalgebra $K_{\mathrm{alg}}\langle X\rangle$, and we have the inclusions:

$$K\langle X\rangle \subseteq K_{\mathrm{rat}}\langle X\rangle \subseteq K_{\mathrm{alg}}\langle X\rangle \subseteq K\langle\langle X\rangle\rangle.$$

(ii). The rules of a given grammar, which were interpreted as monoid equations in §5, can now be interpreted as equations in the power series ring. Thus let G be a grammar on X; with each word w on X we associate a non-negative integer a_w, the number of ways of deriving w in G. The power series

$$f = \sum a_w w \quad \text{in} \quad \mathbf{Z}\langle\langle X\rangle\rangle \tag{4}$$

describes the resulting language completely, in fact it takes multiplicities into account. The language $L(G)$ is just the *support* of f. In (4) all the coefficients are non-negative, but we can interpret a series with coefficients of both signs as the *difference* of two languages. The operations of union, product and * occurring in Theorem 2.1 correspond to sum, product and quasi-inverse: $x^* = 1 + x + x^2 + \ldots = (1-x)^{-1}$.

To give an example, the language xy^* has rules $\sigma \to x$, $\sigma \to \sigma y$. These correspond to the monoid equation $L = x + Ly$; the power series equation is obtained by replacing L by a letter such as u:

$$u = x + uy,$$

and it has the solution $u = x(1-y)^{-1} = \sum xy^n$. Clearly this series is rational and it is not hard to see that every regular grammar gives rise to a rational series with non-negative coefficients (cf. Theorem 2.1).

To state the converse we need the notion of a *semiring*. This is a monoid under addition and multiplication, linked by the distributive laws, thus it only fails to be a ring because subtraction is missing. E.g. the non-negative integers $\mathbf{N}$ form a semiring, and so does the set $\mathbf{N}\langle\langle X\rangle\rangle$ of formal power series over $\mathbf{N}$. Rational and algebraic power series are defined as before. Now we have (Kleene [56]):

THEOREM 6.1. *A language L is rational if and only if it is the support of a rational power series in $\mathbf{N}\langle\langle X\rangle\rangle$.*

Similarly the algebraic power series correspond to CF-languages (Schützenberger [62]):

THEOREM 6.2. *A language L is context-free if and only if it is the support of an algebraic power series in $\mathbf{N}\langle\langle X\rangle\rangle$.*

Thus the series $u = \sum x^n y^n$ is algebraic, for it is of the form $u = v+1$, where

$$v = xvy+xy.$$

That the series $\sum x^n y^n$ is not rational can be seen by inspecting its Hankel matrix, soon to be defined.

(iii). Let M be a direct product of free monoids; we can again form the monoid algebra $R = K[M]$ of M over K, and its completion by formal power series, $\hat{R}$. Rational power series may be defined as before, as components of solutions of equations of the type (1). A series f is said to be *recognizable*, if there is a matrix representation $\mu : M \to K_n$ and a matrix $P \in K_n$ such that

$$f = \sum tr(P.\mu(w))w, \tag{5}$$

or equivalently, if there is a matrix representation $\mu : M \to K_n$ and $\gamma \in K$ such that $f-\gamma$ is the $(1,n)$-entry of $\sum\mu(w)w$ (Schützenberger [**62**], Fliess [**70**]). It can then be shown that a subset of M is recognizable (or rational) if and only if it is the support of a recognizable (or rational) series with coefficients 0 or 1.

To obtain a connexion between rational and recognizable series, corresponding to Theorem 5.3, suppose first that M is free on $x_1, \ldots, x_m$. Then

$$\begin{aligned}\sum\mu(w)w &= I+\sum\mu(x_i)x_i+\sum\mu(x_i x_j)\, x_i x_j+ \ldots \\ &= I+A+A^2+ \ldots = (I-A)^{-1},\end{aligned}$$

where $A = \sum\mu(x_i)\, x_i$. Thus a recognizable series in $K\langle\langle X\rangle\rangle$ is rational, and conversely, because A in (1) can always be taken to be linear in the x_i. This is the process of linearization by enlargement (" Higman's trick "): to get rid of a product ab, in the (n, n)-position say, we take the diagonal sum with 1 and apply elementary transformations:

$$\begin{pmatrix} c+ab & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} c+ab & a \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} c & a \\ -b & 1 \end{pmatrix}$$

(where only the bottom right-hand corner is shown).

Next let $M = M' \times M''$, where M' is free on $x_1, \ldots, x_r$ and M'' free on $y_1, \ldots, y_s$. Then

$$\begin{aligned}\sum\mu(w)w = I+\sum\mu(x_i)\, x_i+\sum\mu(y_p)\, y_p+\sum\mu(x_i x_j)\, x_i x_j+\sum\mu(x_i y_p)\, x_i y_p \\ +\sum\mu(y_p y_q)\, y_p y_q+ \ldots \\ = \left(I+\sum\mu(x_i)\, x_i+(\sum\mu(x_i)\, x_i)^2+ \ldots\right)\left(I+\sum\mu(y_p)\, y_p+(\sum\mu(y_p)\, y_p)^2 \ldots\right) \\ = (I-A)^{-1}\,(I-B)^{-1},\end{aligned}$$

where $A = \sum\mu(x_i)\, x_i$, $B = \sum\mu(y_p)\, y_p$. These equations show very clearly the crucial effect of commutativity on the form the series $\sum\mu(w)w$ can take. The result (cf. Fliess [**70″**]) may be summed up as

THEOREM 6.3. *Let $X = \{x_i\}$ and $Y = \{y_p\}$ be two sets of non-commuting indeterminates, where the x's commute with the y's. Then the recognizable series in the x's and y's constitute the tensor product $K_{\text{rat}}\langle X\rangle \otimes K_{\text{rat}}\langle Y\rangle$.*

Thus a recognizable series in two commuting variables x, y has the form

$$f(x, y)/g(x)\,h(y),$$

where f, g, h are polynomials and $g(0) \neq 0, h(0) \neq 0$.

Every recognizable series is rational, but not conversely, e.g. if x, y commute, $(1-xy)^{-1}$ is rational but not recognizable. To obtain a general criterion we introduce the Hankel matrix. With any series $f = \sum a_w w$ we associate the infinite matrix whose rows and columns are indexed by the monoid M:

$$H(f) = (a_{uv}).$$

This is the *Hankel matrix* of f; its rank is defined in the obvious way as the supremum of the ranks of its finite submatrices. Now one can prove (Schützenberger **[61']**, Carlyle–Paz **[71]**, Fliess **[74]**):

THEOREM 6.4. *Let R be the formal power series ring of a direct product M of free monoids, with coefficients in a field. Then $f \in R$ is recognizable if and only if its Hankel matrix $H(f)$ has finite rank. If the rank is N, then the matrices in* (5) *may be taken to be $N \times N$.*

There is also an analogue of the pumping lemma for recognizable series (Fliess **[71]**):

THEOREM 6.5. *Let M, R be as in Theorem* 6.4 *and let $f = \sum a_w w \in R$ be recognizable but not a polynomial. Then there exist words $u, v, w \in M$, $v \neq 1$ such that the series in the central indeterminate t:*

$$\sum a_{uv^n w} t^n$$

is rational (*i.e. in $K(t)$*) *but not a polynomial in t.*

As we have seen, when M is free, rational and recognizable series are the same, and either of Theorems 6.4, 6.5 may be used to show that $\sum x^n y^n$ is not rational.

With every subset P of a monoid we can define a Hankel matrix $H(f)$, where f is the characteristic series of P: $f = \sum a_w w$, where $a_w = 1$ or 0 according as w is in P or not. Then $H(f)$ has finite rank if and only if P is recognizable (Fliess **[72]**).

(iv). As an example of an algebraic series consider the restricted Dyck language on $x_1, \ldots, x_m, x'_1, \ldots, x'_m$ (with $x_i x'_i \mapsto 1$). This has the equation

$$u = 1 + \textstyle\sum_i u x_i u x'_i.$$

Similarly, the Łukasiewicz language on $x_0, \ldots, x_m$ is defined by

$$u = x_0 + u x_1 + \ldots + u^m x_m.$$

The following necessary condition for an algebraic series, based on a theorem of Eisenstein (Pólya-Szegö **[25]**) is often useful (Fliess **[72]**):

THEOREM 6.6. *Let A be an integral domain with field of fractions K. Given any algebraic series f in $K\langle\langle X\rangle\rangle$, there exists $m \in A$, $m \neq 0$ such that on substituting mx in f for each $x \in X$ we obtain an algebraic series in $A\langle\langle X\rangle\rangle$.*

The proof is as in the classical case: in the equations for f we can make the substitution $x \mapsto mx$ so that all coefficients come to lie in A. E.g. the criterion shows that $\sum x^n/2^{n^2}$ is not algebraic.

(v). In order to describe the intersection of languages we introduce the following operation on power series. With $f = \sum a_u u$, $g = \sum b_u u$ we associate the series

$$f \odot g = \sum c_u u, \quad \text{where} \quad c_u = a_u b_u.$$

This operation is of course well known in analysis, where it is called the *Hadamard product*, and this name is used here too. If f, g have supports L, M respectively, then $f \odot g$ has support $L \cap M$. Clearly the Hadamard product is commutative, associative and distributive over addition. E. Borel observed that the Hadamard product of two rational series in one variable is again rational, but this does not extend to two commuting indeterminates, as the following example due to Hurwitz [**99**] shows;

$$f = \sum x^n y^n = (1-xy)^{-1},$$

$$g = \sum \binom{m+n}{n} x^m y^n = (1-x-y)^{-1},$$

$$f \odot g = \sum \binom{2n}{n} x^n y^n = (1-4xy)^{-1/2}.$$

For any ring R, denote the $n \times n$ matrix ring over R by $\mathfrak{M}_n(R)$ or also R_n, then it is not hard to see that

$$\mathfrak{M}_n(K_x[M]) = \mathfrak{M}_n(K)_x\,[M],$$

where M is a direct product of free monoids, and x = rat, alg or rec (= recognizable) If $\mu : X^* \to K_n$ is a representation and we substitute in any element of $K_x[M]$, $\mu(x)x$ for x, we obtain a series in $\mathfrak{M}_n(K_x[M])$. In this way we find (Schützenberger [**62**])

THEOREM 6.7. *In the power series ring over a direct product of free monoids, the Hadamard product of a rational (or recognizable or algebraic) series and a recognizable series is again rational (or recognizable or algebraic).*

In particular, applied to $\mathbf{Z}\langle\langle X \rangle\rangle$ this yields another proof that the intersection of a rational and a CF-language is CF. For series in one variable Theorem 6.7 was first proved by Jungen [**31**], who also observed that the Hadamard product of two algebraic series need not be algebraic:

$$f = \sum \binom{2n}{n} x^n = (1-4x)^{-1/2},$$

$$f \odot f = \sum \binom{2n}{n}^2 x^n = \frac{2}{\pi} \int_0^{\pi/2} \frac{d\theta}{\sqrt{(1-4x.\sin^2\theta)}} = \frac{2}{\pi} \int_0^1 \frac{dt}{\sqrt{((1-t^2)(1-4xt^2))}}.$$

However, over a perfect field of characteristic $p \neq 0$, the Hadamard product of two algebraic series is algebraic (Furstenberg [**67**], Fliess [**72**]).

(vi). The free algebra and the free power series ring over a field have been studied from various aspects (Cohn [71]), and it is natural to examine their relation to $K_{\mathrm{rat}}\langle X\rangle$ and $K_{\mathrm{alg}}\langle X\rangle$ more closely. In $K\langle\langle X\rangle\rangle$ we can equate cofactors of the generators x_i, or more generally, if the elements $u_i \in X^*$ are such that none is a right multiple of any other, then $\sum u_i f_i = \sum u_i g_i$ implies $f_i = g_i$. Hence the operation $\sum u_i f_i \mapsto f_v$ (for any v) is well-defined; it is called a *right transduction*, and left transductions are defined similarly. These operations correspond to the transductions defined in §4. A transduction maps each rational or algebraic series to a series of the same type (Fliess [72]), and this has many interesting consequences. To describe them we need some definitions.

A ring R is called a *free right ideal ring* or *right fir* for short, if every right ideal of R is free of unique rank. *Left firs* are defined similarly and a left and right fir is called a *fir*. For commutative rings a fir is just a principal ideal domain, and in fact any principal ideal domain, commutative or not, is a fir, but the class of firs is much wider than this, e.g. any free algebra over a field is a fir. By a *semifir* one understands a ring in which any finitely generated right ideal (or equivalently, every finitely generated left ideal) is free, of unique rank. Commutative semifirs are just Bezout rings; to give a general example, the free power series ring over a field, $K\langle\langle X\rangle\rangle$ is a semifir, but not a fir, unless X reduces to a single element.

A ring extension $R \subseteq S$ is said to be *inert* if any factorization of an element of R in S can be reduced to one in R, i.e. given $c \in R$, if $c = ab$ $(a, b \in S)$, then there exists a unit u in S such that $au, u^{-1}b \in R$. Let M be a right R-module; a family (u_i) of elements of M is said to be (linearly) *independent* over R if $\sum u_i a_i = 0$ $(a_i \in R)$ implies $a_i = 0$ for all i. The family is called *weakly independent* over R if no u_i lies in the R-module generated by the others. Clearly any independent family is weakly independent, but the converse need not hold. Now if S is a semifir and R is a subring such that any weakly independent family in S, qua right R-module, is independent, then it is easily shown that R is again a semifir and is inert in S. This condition is easily verified for the pair $K_{\mathrm{rat}}\langle X\rangle$, $K\langle\langle X\rangle\rangle$, where K is a field, using left transductions, hence $K_{\mathrm{rat}}\langle X\rangle$ is a semifir, inert in $K\langle\langle X\rangle\rangle$. Corresponding results hold for $K_{\mathrm{alg}}\langle X\rangle$. The ring $K_{\mathrm{alg}}\langle X\rangle$ is not a fir (Cohn [71], Exercise 5.8.6, p. 209). (27.11.74) W. J. Dicks has just proved that $K_{\mathrm{rat}}\langle X\rangle$ is a fir.

The ring $K\langle\langle X\rangle\rangle$ has rather simple factorization properties, when K is a field: it is a *rigid UFD* (Cohn [62'], [71], Chapter 3). This means that every element not 0 or a unit can be written as a product of atoms (i.e. unfactorable elements): $a = p_1 \dots p_r$ and if $a = q_1 \dots q_s$ is another such factorization, then $r = s$ and $q_i u_i = u_{i-1} p_i$ $(i = 1, \dots, r)$, where the u's are units and $u_0 = u_r = 1$. Let S be a rigid UFD and R an inert subring such that any non-unit of R is a non-unit in S; then R is again a rigid UFD. It follows that both $K_{\mathrm{rat}}\langle X\rangle$ and $K_{\mathrm{alg}}\langle X\rangle$ are rigid UFDs (Fliess [72]).

7. *Transformational grammars*

We have already seen that the classes of formal languages described earlier fail to do justice to natural languages: context-sensitive languages are too wide, while

context-free languages are too restrictive. Here is another very simple illustration of the same fact. Many natural languages—including English—use two alphabets, capitals and lower case letters, and it is a rule that the first letter in each sentence is a capital. It is not at all obvious how to accommodate this type of rule in a formal grammar. To give a further example, many sentences occur in several different forms, such as active and passive; e.g. consider the sentences:

1. The cat ate the mouse.
2. The mouse ate the cat.
3. The mouse was eaten by the cat.
4. The cat was eaten by the mouse.

A moment's glance shows that 1 and 3 mean the same thing, as do 2 and 4, but *not* 1 and 2, nor 1 and 4. Again there is no mechanism (within the frame-work of phrase-structure grammars) to take account of these facts. This difficulty was recognized by Chomsky, who proposed the notion of transformation rules. Roughly speaking, instead of merely providing rules of sentence formation, one also has rules of sentence transformation. Without entering into the details, we can easily see how this might apply to the above examples. A language might have the transformation rule which replaces the first letter of a sentence by the corresponding capital, and we only accept sentences where this rule has been applied. Similarly one can formulate rules for turning an active form into the passive, and so obtain from each sentence (if expressed in active form) another one (in passive form).

The study of transformations (transformational grammar) is less developed than that of phrase-structure grammars. It also has less of an algebraic character, which is why we do not discuss it in greater detail. A serious difficulty is that transformation rules applied to context-free grammars can lead to all languages of type 0 (Peters and Ritchie [**71**]). This means that the transformation rules are not sufficiently restrictive. However, one would hope that a comparison with the rules of natural grammars will eventually lead to a more accurate picture; for an attempt in this direction see Ginsburg–Partee [**69**].

References

Arbib, M. A., 69. *Theories of abstract automata*, (Prentice-Hall, Englewood Cliffs, 1969).

Bar-Hillel, Y., Perles, M. and Shamir, E., 61. On formal properties of simple phrase structure grammars, *Z. Phonetik, Sprachwiss. Kommunikat.*, 14 (1961) 143–172. Reprinted in Handbook of Math. Psychology, vol. 3, (J. Wiley, New York, 1965)

Booth, T L., 67. *Sequential machines and automata theory*, (J. Wiley, New York, London, Sydney 1967).

Cantor, D. G., 62. On the ambiguity problem of Backus systems, *J. Assoc. Comput. Mach.*, 9 (1962), 477–479.

Carlyle, J. W. and Paz, A., 71. Realizations by stochastic finite automata, *J. Comput. System Sci.*, 5 (1971), 26–40.

Chomsky, N., 59. On certain formal properties of grammars, *Information and Control*, 2 (1959), 137–167.

———, 59′. A note on phrase structure grammars, *Information and Control*, 2 (1959), 393–395.
———, 64. *Aspects of the theory of syntax*, (MIT Press, Cambridge, Mass. 1964).
——— and Miller, G. A., 58. Finite state languages, *Information and Control*, 1 (1958), 91–112.
——— and Schützenberger, M.-P., 63. The algebraic theory of context-free languages, in " *Computer programming and formal systems* " (ed. P. Braffort and D. Hirschberg) (North Holland, Amsterdam, 1963).
Claus, V., 71. *Stochastische Automaten*, (B. G. Teubner, Stuttgart, 1971).
Cohn, P. M., 62. On subsemigroups of free semigroups, *Proc. Amer. Math. Soc.*, 13 (1962), 347–351.
———, 62′. Factorization in non-commutative power series rings, *Proc. Cambridge Philos. Soc.*, 58 (1962), 452–464.
———, 65. *Universal Algebra*, (Harper and Row, New York, London, Tokyo, 1965).
———, 70. *Pure and applied algebra*, Inaugural lecture, Bedford College (London, 1970).
———, 71. *Free rings and their relations*, No. 2, LMS monographs, (Academic Press, London, New York, 1971).
———, 71′. The embedding of firs in skew fields, *Proc. London Math. Soc.*, (3) 23 (1971), 193–213.
———, 74. Progress in free associative algebras, *Israel J. Math.*, 19 (1974), 109–151.
Conway, J. H., 71. *Regular algebra and finite machines*, (Chapman and Hall, London, 1971).
Davis, M., 58. *Computability and unsolvability*, (McGraw–Hill, New York, 1958).
Eilenberg, S., 74. *Automata, Languages and Machines*, vol. A, (Academic Press, New York, 1974).
Fliess, M., 70. Inertie et rigidité des séries rationnelles et algébriques, *C. R. Acad. Sci. Paris, Sér. A*, 270 (1970), 221–223.
———, 70′. Transductions algébriques, RIRO R-1 (1970), 109–125.
———, 70″. Séries reconnaissables, rationnelles et algébriques, *Bull. Sci. Math.*, 94 (1970), 231–239.
———, 71. Deux applications de la représentation matricielle d'une série rationelle non-commutative, *J. Algebra*, 19 (1971), 344–353.
———, 72. *Sur certaines familles de series formelles*, Thèse (University of Paris VII, 1972).
———, 75. Matrices de Hankel, to appear.
———, 75′. Séries rationelles positives et processus stochastiques, *Ann. Inst. H. Poincaré, Sect. B.*
Furstenberg, H., 67. Algebraic function fields over finite fields, *J. Algebra*, 7 (1967), 271–277.
Ginsburg, S., 66. *The mathematical theory of context-free languages*, (McGraw–Hill, New York, 1966).
——— and Partee, B., 69. A mathematical model of transformational grammars, *Information and Control*, 15 (1969), 297–334.
Gross, M., 72. *Mathematical models in linguistics*, (Prentice–Hall, Englewood Cliffs, 1972).
——— and Lentin, J., 70. *Introduction to formal grammars*, (Springer–Verlag, Berlin, Heidelberg, New York, 1970).
Harris, Z. S., 51. *Structural linguistics* (formerly entitled: Methods in structural linguistics) (Univ. of Chicago Press, Chicago, 1951, 1960).
Hartmanis, J. and Stearns, R. E., 66. *Algebraic structure theory of sequential machines*, (Prentice–Hall, Englewood Cliffs, 1966).
Hopcroft, J. E. and Ullman, J. D., 69. *Formal languages and their relation to automata*, (Addison–Wesley, Reading, Mass., 1969).
Hurwitz, A., 99. Sur un théorème de M. Hadamard, *C. R. Acad. Sci. Paris*, 128 (1899), 350–353, reprinted in Ges. Werke I, 482–484.
Jungen, R., 31. Sur les séries de Taylor n'ayant que des singularités algébrico-logarithmiques sur leur cercle de convergence, *Comment. Math. Helv.*, 3 (1931), 266–306.
Kleene, S. C., 56. Representation of events in nerve sets, in " *Automata studies* " (ed. C. E. Shannon and J. McCarthy), (Princeton Univ. Press, Princeton, 1956), 3–40.
Kuroda, S. Y., 64. Classes of language and linear-bounded automata, *Information and Control*, 7 (1964), 207–233.

Landweber, P. S., 63. Three theorems on phrase structure grammars of type 1, *Information and Control*, 6 (1963), 131–136.

Myhill, J., 57. *Finite automata and the representation of events*, Wright Air Development Command Tech. Report No. 57–624 (1957), 112–137.

———, 60. *Linear bounded automata*, Wright Air Development Command Tech. Report No. 60–165 (1960).

Nerode, A., 58. Linear automaton transformations, *Proc. Amer. Math. Soc.*, 9 (1958), 541–544.

Nivat, M., 68. Transduction des langages de Chomsky, *Ann. Inst. Fourier* (Grenoble) 18, 1 (1968), 339–456.

———, 70. *Séries rationnelles et algébriques en variables non-commutatives*, Cours du DEA 1969-70.

Parikh, R., 61. Language generating devices, MIT RLE Quarterly Progress Report 60 (1961) 199–212. Reprinted as " On context-free languages " *J. Assoc. Comput. Mach.*, 13 (1966), 570–581.

Paz, A., 66. Some aspects of probabilistic automata, *Information and Control*, 9 (1966), 26–60.

Peters, P. S. and Ritchie, R. W., 71. On restricting the base component of transformational grammars, *Information and Control*, 18 (1971), 483–501.

Pólya, G. and Szegö, G., 25. *Aufgaben und Lehrsätze aus der Analysis II*, (Springer–Verlag, Berlin, 1925) reprinted by (Dover, New York, 1945).

Schützenberger, M.-P., 61. A remark on finite transducers, *Information and Control*, 4 (1961), 185–196.

———, 61′. On the definition of a family of automata, *Information and Control*, 4 (1961), 245–270.

———, 62. On a theorem of Jungen, *Proc. Amer. Math. Soc.*, 13 (1962), 885–890.

Shamir, E., 67. A representation theorem for algebraic and context-free power series in non-commuting variables, *Information and Control*, 11 (1967), 239–254.

Teissier, M., 51. Sur les équivalences régulières dans les demi-groupes, *C. R. Acad. Sci. Paris*, 232 (1951), 1987–1989.

Turakainen, P., 69. Generalized automata and stochastic languages, *Proc. Amer. Math. Soc.*, 21 (1969), 303–309.

Bedford College,
Regent's Park,
London NW1 4NS.