AD 744 206

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Project MAC
Massachusetts Institute of Technology

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
None

3. REPORT TITLE
FURTHER RESULTS ON HIERARCHIES OF CANONIC SYSTEMS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*
Robert Mandl

6. REPORT DATE
June 1972

7a. TOTAL NO. OF PAGES
84

7b. NO. OF REFS
22

8a. CONTRACT OR GRANT NO.
N00014-70-A-0362-0001

b. PROJECT NO. N/A

c. N/A

d. N/A

9a. ORIGINATOR'S REPORT NUMBER(S)
MAC TR-100

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES
None

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research

13. ABSTRACT This thesis outlines a new way of presenting the theory of canonic systems, including a distinction (for methodic reasons) between simple canonic systems and general canonic systems, and proves a series of results on hierarchies of canonic systems. After a brief summary of Doyle's results on a partial hierarchy of canonic systems, a new hierarchy is developed (Chapter II) which relates the general canonic systems not only to all 4 types of formal grammars defined by Chomsky but also to any class of formal grammars definable in terms of productions. It is also shown (Chapter III) that all attempts to define a mathematical system which exactly corresponds to the recursive sets are necessarily fruitless. Doyle's work on how to define "noncontracting canonic systems with predicates of degree 2" (NCST) is continued arriving at a workable definition which permits us to prove [NCST] = [Type 1] (Chpt.4), a conjecture put forth at the 3rd Princeton Conference on Information Sciences and Systems. This result transforms Doyle's hierarchy from "the union of two half-hierarchies and a dangling term (the NCST)" into a complete hierarchy of canonic systems (all 4 types represented). However, this hierarchy is heterogenous: canonic systems corresponding to grammars of types 3 and 2 use only predicates of degree 1, while canonic systems corresponding to grammars of types 1 and 0 use also predicates of degree 2; moreover, not all of them are simple canonic systems. A [homogenous] hierarchy of simple canonic systems with predicates of degree 1 is presented in Chpt.4. Several new classes of canonic systems (non-crossreferencing, non-inserting, and pure canonic systems) are introduced in Chapter 6, where their properties are explored, and a classification schema and several hierarchies are developed.

DD FORM 1473 (PAGE 1)
S/N 0102-014-6600

UNCLASSIFIED
Security Classification

2a

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
canonic system alphabet context sensitive subrecursive						

2b

MAC TR-100

FURTHER RESULTS ON HIERARCHIES OF CANONIC SYSTEMS

ROBERT MANDL

MAY 1972

This research was supported by the
Advanced Research Projects Agency of
the Department of Defense under ARPA
Order No. 433, and was monitored by
ONR under Contract No. N00014-70-A-0362-0001

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

CAMBRIDGE MASSACHUSETTS 02139

ACKNOWLEDGEMENT

I wish to take this opportunity to express my thanks to Prof. John Donovan, who took the time from a busy schedule to supervise this thesis and whose initial work on canonic systems provided the motivation for this research.

I should like to express my appreciation to Amitava Bagchi, who contributed significantly to the successful completion of this thesis by critically reading several successive versions of my work on canonic systems.

I wish to thank Prof. Malcolm Jones for helping me find the right set of goal priorities during a very busy summer.

Finally, I am grateful to Project MAC, which provided the facilities and support for this thesis and a stimulating environment for formal research. Particular thanks go to several individuals at Project MAC, especially to Joseph Haggerty, Norman Kohn, and Hoo-min Toong, for their interest and comments during many discussions on the thesis subject.

Cambridge, Massachusetts
August 1969

Robert Mandl

TABLE OF CONTENTS

Page

ABSTRACT 2

ACKNOWLEDGEMENT 3

Chapter I : Simple and general canonic systems 5

Chapter II : A hierarchy of general canonic systems 23

Chapter III : Subrecursive classes of canonic systems 35

Chapter IV : Canonic systems for context-sensitive sets 39

Chapter V : Further hierarchies of canonic systems 55

Chapter VI : Non-crossreferencing, simple, and non-inserting canonic systems. Classification of canonic systems 64

REFERENCES 80

LIST OF DEFINITIONS 82

LIST OF THEOREMS 83

LIST OF FIGURES 84

CHAPTER I

SIMPLE AND GENERAL CANONIC SYSTEMS

This chapter presents the differences between the traditional definitions and the ones we will use, and builds the theory of canonic systems according to the new specifications. It also includes the motivation for the reorganization of canonic systems.

Canonic systems were first defined in Donovan 1966 . The starting point of our work was the version presented in Donovan and Doyle 1968, pp. 3-9. The reader is assumed to be acquainted with this work, and therefore we will not repeat that definition but rather present the modifications we have introduced and the arguments behind them, and then present only the modified definition.

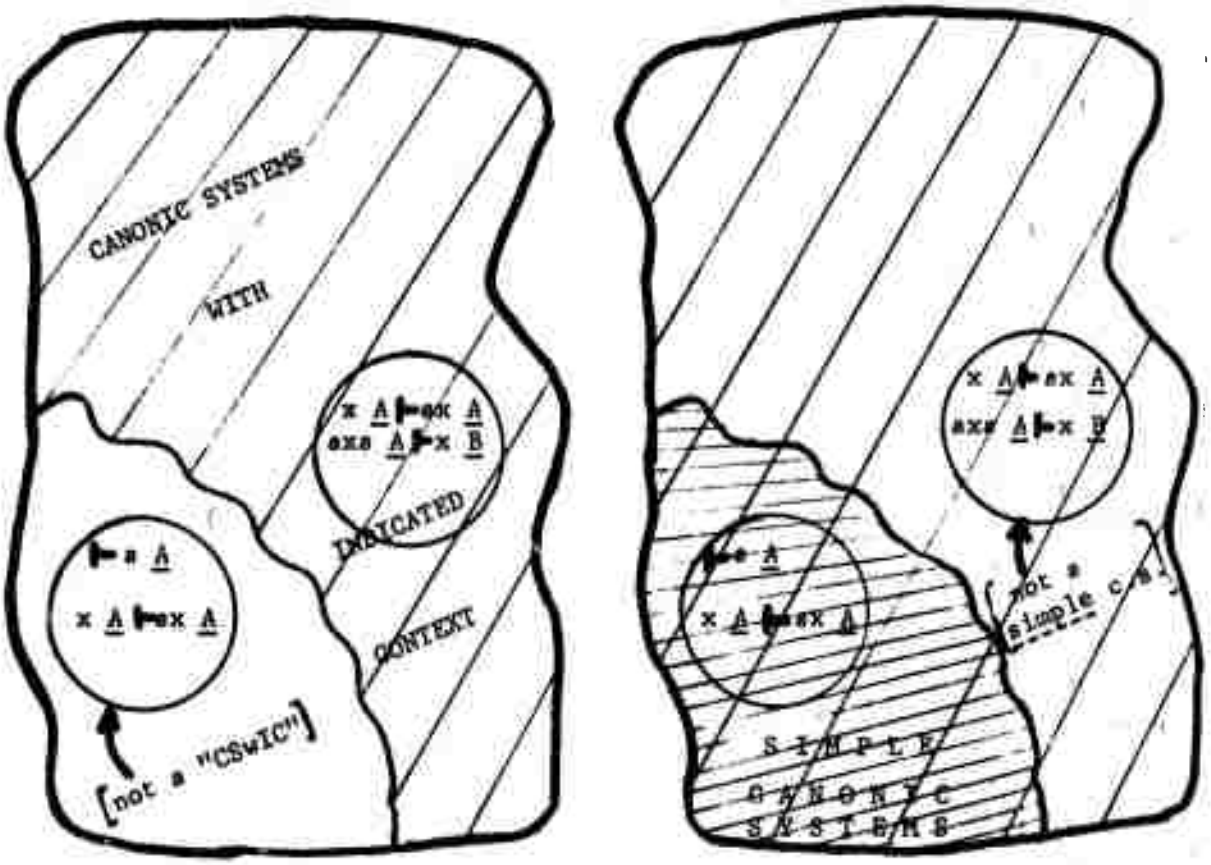
A canon used to be defined as a list of statements followed by the sign ⊢ and then followed by a statement, where a statement (traditionally called 'remark') is composed of a term of some degree followed by a predicate of the same degree. A term of degree n is an n-tuple of arbitrary concatenations of variables and words on the given alphabet, the words surrounding the variables being referred to as the context of the variables. A particular case was singled out, the case when context is actually indicated, and the canonic systems satisfying this condition, i.e. canonic systems which contain at least one canon in which there is an instance of variables and symbols concatenated together in the same term, were called canonic systems with indicated context (CSwIC) [Donovan and Doyle 1968, p. 28; Haggerty 1969, p. 41], but not

much was known about them beyond the observation that they appear to be rather powerful. Most classes of canonic systems encountered in the course of research were not "canonic systems with indicated context" in the sense of the old definition mentioned above; moreover, in all cases but one, constructive proofs for the existence of canonic systems with a certain property yielded canonic systems which were not "with indicated context", and the same holds for Alsop's "canonic translator" [Alsop 1967] . Because of these, and especially in view of Haggerty's recent result [Haggerty 1969] that contextual indications can be dispensed with, we have decided not to regard as a distinguished class the class of canonic systems which do exercise the option of indicating context, but rather to distinguish the class of canonic systems which have no such option available, and call them simple canonic systems , while the unrestricted canonic systems will sometimes be referred to, for emphasis only, as general canonic systems or as canonic systems with indicated context . [Therefore the new meaning of this term is that we have the option of indicating contextual conditions, and nothing more than the option, in contrast with the old meaning, which required us to exercise this option in at least one canon.]

The situation is similar to that encountered in automata theory, in connection with the definition of nondeterministic automata. The old definition of canonic systems with indicated context corresponds to the following hypothetical definition of the concept 'nondeterministic Turing machine' [NTM] : "A NTM is a TM in which there is at least one state satisfying the condition that for at least one symbol of the tape alphabet there are two or more quadruples [or quintuples, if we work with quintuples] in the specification of the TM" . According to this definition, deterministic TM were not particular cases of NTM but constituted a class disjoint from the class of NTM. Clearly, this choice of a definition

OLD

NEW



CANONIC SYSTEMS

CANONIC SYSTEMS =
= GENERAL CANONIC SYSTEMS =
= C.S. WITH INDICATED CONTEXT

Figure 1

Graphic representation of the changes in terminology

The circles represent particular examples of canonic systems.

would not be fortunate, and, in fact, this is not the definition of nondeterministic Turing machines, as everybody knows; rather, the deterministic TMs were singled out (were distinguished) as a particular case of NTMs. The new definition of canonic systems with indicated context and the introduction of the simple canonic systems were necessitated in order to "normalize" the

usage in canonic systems, to switch from a nomenclature corresponding to the hypothetical definition of NTM, in our example, to a nomenclature which corresponds to the true definition of NTM.

Similarly, instead of talking of canonic systems with insertion, or of canonic systems with crossreferencing, etc., we would single out the canonic systems without insertion, or without crossreferencing, etc.. These classes of canonic systems will be introduced and studied in Chapter VI.

The way in which we chose to "implement" this reorganization is by introducing p-terms ("premise terms") and their lists along with terms and their lists, and premises along with statements . A p-term is an n-tuple each of whose elements is a "pure" concatenation (containing either exclusively variables or exclusively symbols). This, incidentally, also eliminates the recursion on term , so that it will no longer the case that a substring of a term is, automatically, itself a term.

We are now ready to present the definition of simple canonic systems.

Definition 1. A simple canonic system (of level i) is a septuple

$$\mathcal{C}_i = (C_i , V_i , M_i , P_i , S_i , D_i , \mathcal{C}_{i-1})$$

where

C_i is a finite set of canons (rules of inference);

V_i is the alphabet used to form the strings generated by $\mathcal{C}_i$;

M_i is a finite set of variables used to stand for elements of any predicate ;

P_i is a finite set of predicates used to name sets of tuples. The number of components in the tuples is the degree .

S_i is a finite set of punctuation signs;

D_i ($\subseteq P_i$) is a set of sentence predicates whose union will be defined to be the language specified by the canonic system.

ζ_{i-1} is the "object" canonic system.

This definition is not complete until we say what the canons, variables, predicates are and what we can do with them.

However, since the reader is assumed to be familiar with these concepts, these will not be repeated here. Most of the differences have been outlined above, and a formal definition, using second-level canonic systems, will now be given. The reader is urged to compare it with the old definition of canonic systems [Donovan and Doyle 1968, pp. 10-18], to get a complete and accurate image of the changes that were introduced. In order to facilitate the comparison, our exposition will also be given by way of an example, and will use the same example, a canonic system defining the set of numbers composed of the digits 1,2 and 3. Moreover, the drawing on page 5 of the above-mentioned work is presented below in a updated form as Figure 2 to provide a quasi-pictorial representation of some of the changes introduced. General canonic systems are defined similarly, but allowing arbitrary concatenations of variables and symbols not only in the conclusion but also in the premises.

$$\mathcal{G}_1 = (C_1 , V_1 , M_1 , P_1 , S_1 , D_1 , \mathcal{G}_0)$$

where C_1 :

$\vdash$ 1 digit

$\vdash$ 2 digit

$\vdash$ 3 digit

x digit $\vdash$ x number

x digit ; y number $\vdash$ yx number

$V_1 = \{ 1 , 2 , 3 \}$

$M_1 = \{ x , y \}$

$P_1 = \{$ digit, number $\}$

$S_1 = \{ ; , \vdash \}$

$D_1 = \{$ number $\}$

$\mathcal{G}_0 = (\mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B}, \mathcal{B})$.

The following parse of the fifth canon of this system illustrates the metalanguage used to describe canons.

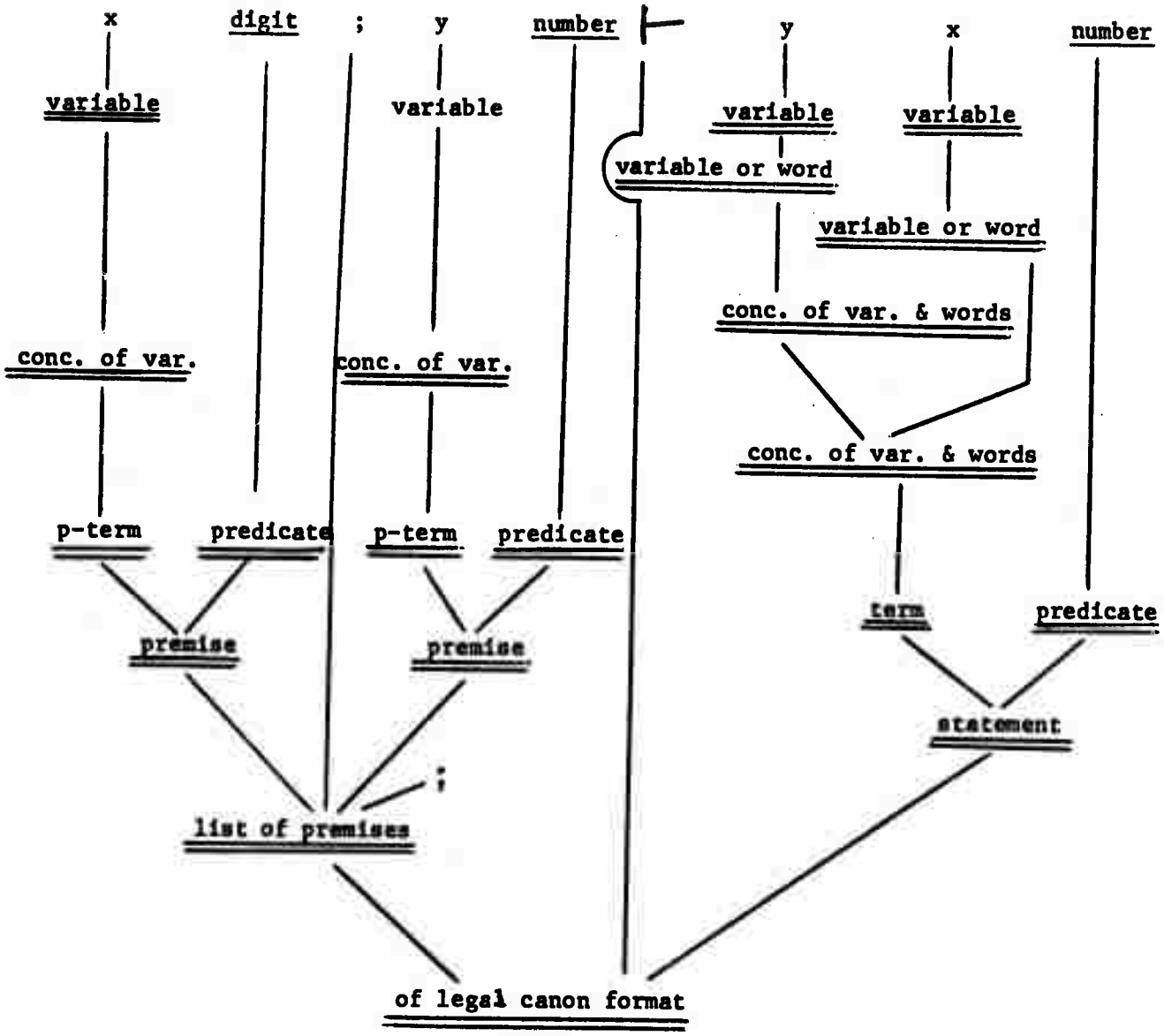

x
digit
;
y
number
y
x
number
variable
variable
variable
variable
variable or word
variable or word
conc. of var.
conc. of var.
conc. of var. & words
conc. of var. & words
p-term
predicate
p-term
predicate
term
predicate
premise
premise
statement
;
list of premises
of legal canon format

Figure 2

The second-level canonic system is a 7-tuple

$$\mathfrak{C}_2 = (C_2,\ V_2,\ M_2,\ P_2,\ S_2,\ D_2,\ \mathfrak{C}_1)$$

where

C_2 = { the canons listed on the following pages }

V_2 = { 1, 2, 3, digit, number, x, y, ; , $\vdash$ }

M_2 = { q, r, s, t, u, v, w }

P_2 = { predicates as defined in the canons }

S_2 = { ;; , $\models$, $<$, $>$, $\prec$ }

D_2 = { legally defined string }

$\mathfrak{C}_1$ is the first-level canonic system

The canons of the second level must formally define the metalanguage and operations of the first level; these canons are presented on the following pages with a brief discussion of the motivation and use of some of the canons. The particular manner in which we have constructed the second-level canons system allows this system to define other canonic systems with only slight modifications, which include, mainly, canons which define the set of canons of the system being defined.

(1.1) $\vdash$ 1 symbol

(1.2) $\vdash$ 2 symbol

(1.3) $\vdash$ 3 symbol

(2.1) $\vdash$; sign

(2.2) $\vdash$ **sign**

(3.1) $\vdash$ x variable

(3.2) $\vdash$ y variable

(4.1) $\vdash$ digit predicate

(4.2) $\vdash$ number predicate

(4.3) $\vdash$ number sentence predicate

(5.1) $\vdash$ λ word (λ is the null string)

(5.2) u symbol ;; v word $\vdash$ uv word

(6.1) u variable $\vdash$ u concat. of var.

(6.2) u concat. of var. ;; v variable $\vdash$ uv concat. of var.

(6.3) u concat. of var. $\vdash$ u p-term

(6.4) u word $\vdash$ u p-term

(6.5) u variable $\vdash$ u concat. of var. & words

(6.6) u word $\vdash$ u concat. of var. & words

(6.7) u concat. of var. & words ;; v variable $\vdash$ uv concat. of var. & words

(6.8) u concat. of var. & words ;; v word $\vdash$ uv concat. of var. & words

(6.9) u concat. of var. & words $\vdash$ u term

(7.1) t p-term ;; u predicate $\vdash$ tu premise

(7.2) t term ;; u predicate $\vdash$ tu statement

(8.1) λ list of premises

(8.2) u list of premises ;; v premise $\vdash$ uv; list of prem.

(8.3) λ list of statements

(8.4) u list of statements ;; v statement ⊨ uv; list of statements

For efficiency's sake, one might add

(7.0) u premise ⊨ u statement

(8.0) u list of premises ⊨ u list of statements

Note especially the intuitive meaning of p-term : a p-term is either a concatenation of variables or a single word (in V*). A term is an arbitrary concatenation of words and variables. The difference between premise and statement is that premise does not allow concatenations of variables and symbols (hence it is "context-free") while statement allows them. One and the same variable may occur several times in the hypothesis and the conclusion of a canon. (This is an instance of crossreferencing.)

(9.1) u word ⊨ u constant

(9.2) u predicate ⊨ u constant

(9.3) u sign ⊨ u constant

(9.4) u constant ;; v constant ⊨ uv constant

(10.1) ⊨ < x < y > differ

(10.2) < u < v > differ ⊨ < v < u > differ

The following canons define a set or ordered quadruples named substitution. They specify the substitution of constants for variables in canons. Thus each canon of the first-level canonic system, if it contains any variables at all, gives rise to a class of

specific instances of canons. These instances are obtained when any terminal string is substituted for the variables in the canon.

Substitution is defined by a 4-tuple $< w < v < s < t >$. The first element, w , is a word; the second element, v , is a variable; the third element is the original nonempty string s ; and the fourth element is the string t which results when the word is substituted for each occurrence of the variable in the original string.

(11.1) w word ;; v variable $\models$ $< w < v < v < w >$ substitution

(11.2) w word ;; s variable ;; v variable ;; $< v < s >$ differ $\models$ $< w < v < s < s >$ substitution

(11.3) w word ;; v variable ;; s constant $\models$ $<w < v < s < s >$ substitution

(11.4) $< w < v < s < q$ substitution ;; $<w < v < x < t \models$ $< w < v < sx < qt >$ substitution

Canon 11.1 defines the substitution of a word for a variable in a string consisting of only that variable. Canon 11.2 defines the substitution of a word for a variable in a string which does not include that variable; this substitution has no effect. Canon 11.3 defines the substitution of a word for a variable in a constant string; this substitution has no effect. Canon 11.4 defines substitution in general.

Canons 12.1 - 12.5 list the canons of the first-level canonic system.

(12.1) $\models \vdash$ 1 digit canon

(12.2) $\models \vdash$ 2 digit canon

(12.3) $\models \vdash$ 3 digit canon

(12.4) $\models$ x digit $\vdash$ x number canon

(12.5) $\models$ x digit ; y number $\vdash$ yx number canon

In order to make sure that indeed the canons are of the required format, we add:

(13.0) v statement $\models \vdash$ v of legal canon format

(13.1) u; list of premises ;; v statement $\models$ u $\vdash$ v of legal canon format

(13.2) u canon ;; u of legal canon format $\models$ u instance of legal canon

(Canon 13.3 defines the set of canons in which constants have been substituted for some or all of the variables.)

(13.3) u instance of legal canon ;; v variable ;; w word ;; $< w_{<} v_{<} u_{<} t >$ substitution $\models$ t instance of legal canon

Canon 13.4 defines a subset of the canons; this subset is the set of all canons which contain only constants. Derivations will be generated from "canons with constants."

(13.4) u instance of legal canon ;; u constant $\models$ u instance with constants

Canons 14, 15.1 and 15.2 define the sets named constituent of and occurrence ; these sets are used in defining derivation. It has been stated that a statement can be derived as the conclusion of a canon by showing that all of the statements in the premise have been derived; i.e., the premise occurs in the derivation. Thus, the meaning of the "occurrence" of a statement in a list of statements must be defined. The concept "occurrence" must be generalized to show that all of the statements in the premise have already occurred in the derivation; this generalization is the set constituent of.

(14) v statement ;; r list of statements ;; t list of statements $\models$ <v $_<$ rv;t> occurrence

(15.1) u list of statements $\models$ <u $_<$ u> constituent of

(15.2) u list of statements ;; v list of statements ;; <u $_<$v> constituent of ;; < w $_<$v> occurrence $\models$ <uw$_<$ v >constituent of

(16.1) λ derivation

(16.2) t derivation ;; w list of statements ;; u statement ;; w$\vdash$u instance with constants ;;<w $_<$t > constituent of $\models$ tu; derivation *

---

* The canon (16.2), which also occurs in the definition of general canonic systems, is not itself admissible in a simple canonic system. In other words, the higher-level canonic systems that we construct here are not themselves simple , whether or not they describe simple canonic systems. However, it will shortly be evident, using a result of Haggerty, that they can be converted to simple canonic systems.

The final set to be defined is the set of strings derived by derivations; each of these strings is simply the last statement in some derivation.

(17) tu; derivation ;; u statement ⊢ u legally derived string.

Canons 16 and 17 are of particular interest since they define the essence of a proof (derivation) and a law (legally derived string) in all mathematical systems.

This completes the construction of the canons of the second-level canonic system. In this example the first-level canonic system had only predicates and terms of degree 1; modification to the second-level system may be made to handle predicates and terms of higher order in the first-level canonic system [Donovan and Doyle 1968] .

The metalanguage describing the second-level canonic system (canon, substitution, derivation, etc.) has not been defined; a third level system would be needed to define it formally. The form of the third-level canonic system is almost identical to that of the second-level system with appropriate changes in notation, i.e. predicates are underlined three times and the punctuation signs are ';;;' and '⊢'. We now outline briefly a formation of a third-level canonic system for this particular second-level system. We remark first that when we specified the second-level canonic system, we set up a standard frame, independent of $\mathfrak{C}_1$ (canons 5,6,7,8,9,10.2,11,13,14,15,16,17) to which $\mathfrak{C}_1$-dependent canons

were added: 1, 2, 3, 6, 10.1, 12. The same precedure will be followed here. The third-level ($(i+1)^{th}$ -level, $i \geq 2$) canonic system may be constructed from the second-level (i^{th}-level) canonic system by the following algorithm:

1. To obtain the $\mathfrak{C}_2$-independent ($\mathfrak{C}_i$-independent) canons, use the standard frame, but make the appropriate changes in notation, i.e. underline the predicates one additional time and add one more semicolon wherever the sign ';;' ($;^i$) occurs.

2. To obtain the $\mathfrak{C}_2$-dependent ($\mathfrak{C}_i$-dependent) canons, use the members of these sets listed in the definition of the second-level (i^{th}-level) canonic system as the terms of the appropriate canons of the second-level canonic system and underline the predicates one additional time.

Thus, the $(i+1)^{th}$-level canonic system can be constructed from the ith-level canonic system with a minimum of effort. Thus, it can be seen that all higher-level canonic systems have the same basic form. Since no level defines its own operations, each level is logically consistent.

For purposes of discussion, at some level the metalanguage of the level must be defined informally. It appears that the second level would be an appropriate level to do this. Recall that, for a given problem, the first-level canonic system defines the problem; the second-level canonic system defines the operation of the first-level canonic system. All higher-level systems define the

operation of previous-level systems. Thus, by selecting the second-level to informally define the metalanguage, the first level canonic system (which defines the problem) is precisely defined and logically consistent.

For the case when the "object" canonic system $\mathfrak{C}_1$ is not a simple canonic system, the following changes will have to be made in the second-level canonic system $\mathfrak{C}_2$ formally specifying "the anatomy and physiology" of $\mathfrak{C}_1$:

1) 6.1-6.4 7.1 8.1 8.2 are unnecessary
(6.5-6.9 7.2 8.3 8.4 alone will do in this case);

2) 13.1-13.2 should be replaced by

(13.1) u; list of statements ;; v statement $\vdash$ u $\vdash$ v
of legal canon format

(13.2) u canon ;; u of legal canon format $\vdash$
u accepted canon

3) Obviously, all the $\mathfrak{C}_1$-dependent canons of $\mathfrak{C}_2$ will be chosen so as to reflect the particular components of $\mathfrak{C}_1$.

Suitable changes may be made to allow for predicates of higher degrees. Examples of canons allowed in general canonic systems are:

x A $\vdash$ ax A

axby A $\vdash$ xy B

xby A $\vdash$ xcyd B

x number; ©x, book descriptor ⊢ x year of copyright

(x,y ∈M; '©', 'a', 'b' , and ', ' are in V)

The sentence symbol (predicate) will be denoted by 'sentence' instead of ' ∑' (D = {sentence}).

The alert reader has undoubtedly noticed another departure from the traditional terminology: our avoidance of the term "terminal alphabet". The set V has been called just plain "alphabet". The reason is that this set does not necessarily correspond to the terminal alphabet of a formal grammar; it may include auxiliary symbols.* In this connection, see also Chapter VI.

Before we study the different hierarchies of canonic systems, we wish to mention several results of Haggerty and to point out one of their implications.

Theorem H-1. Any canonic system can be reduced to one in which no predicate has degree greater than 1 . ["Reduced" means that a statement is provable in the second canonic system iff it is provable in the first one.]

---

* In constructing canonic systems to correspond to regular or to context-free grammars, Doyle took the terminal alphabet of the grammar to serve as alphabet of the canonic system, and the nonterminal alphabet to serve as set of predicates. When, however, he considered grammars of type 0 or 1, using a completely different approach, he correctly used, in fact, the union of the terminal alphabet and the nonterminal alphabet of the grammar to be the alphabet of the resulting canonic system, but he said he included only the terminals. If in his construction the alphabet is to include only the terminal symbols of the grammar, then his construction would not yield a canonic system at all, since some of the "canons" included are of the form ⊢ A nonterminal, where A is neither a symbol nor a variable. Whenever we shall hereafter mention these constructions, we shall assume that the appropriate correction has been made.

[Proof by replacing n-tuples $< s_1 , s_2 , \ldots , s_n >$ by terms of the form $s_1 \$ s_2 \$ \ldots \$ s_n$, where $ is a new symbol, to be used as a separator.]

Theorem H-2. Any canon using indicated context may be reduced to a canon without indicated context (in other words, any canonic system can be reduced to a simple canonic system).

[Proof. Each constant word will be replaced by a variable whose value is specified (by an additional premise) to be in an [adequately defined] singleton set.]

Theorem H-3. Any canonic system can be reduced to one in which each canon has a single premise.

[The proof uses the following basic idea: a canon like

$$\text{term}_1 \ \underline{\text{pred}}_1 \ ; \ \text{term}_2 \ \underline{\text{pred}}_2 \ ; \ \ldots \ ; \ \text{term}_n \ \text{pred}_n \vdash \text{term} \ \underline{\text{pred}}$$

is replaced by

$$< \text{term}_1 , \text{term}_2 , \ldots , \text{term}_n > \ \underline{\text{pred}}_{1,2,\ldots,n} \vdash \text{term} \ \underline{\text{pred}} \ ,$$

where $\underline{\text{pred}}_{1,2,\ldots,n}$ is a new predicate whose degree is the sum of the degrees of $\underline{\text{pred}}_i$, and then additional canons are introduced for the newly-created predicates.]

We remark that, as a consequence of Theorem H-2, the class of simple canonic systems is no less powerful than the class of general canonic systems. Knowing this, one might wonder why bother to defined simple canonic systems if the class of sets definable by them is not different from the class of sets defined by the most general canonic systems. However, the real significance of this theorem is quite different: we should study simple canonic systems precisely because they form a restricted class of simpler canonic systems which still realizes the same computational power. An additional argument is that Alsop's "canonic translator" [Alsop 1967] uses only "simple canons". Moreover, there is nothing to guarantee us that if we apply a certain restriction on the class of all canonic systems and on the class of simple canonic systems, the resulting classes have the same computational power, or that the image of the first restricted class under the transformation of Theorem H-2 is included in the second restricted class.

CHAPTER II

A HIERARCHY OF GENERAL CANONIC SYSTEMS

Canonic systems were first used in specifying the syntax of simulation languages [Donovan 1966], including the features which cannot be expressed in Backus-Naur Form. Since canonic systems, while designed to be more powerful than BNF, were too powerful when first defined (having the full computational power of Turing machines and thus being able to define non-recursive sets), it was felt that restrictions have to be applied so as to render the resulting classes of canonic systems incapable of defining non-recursive sets yet powerful enough to specify the syntax of any programming language. (Experience and intuition have indicated to us that for most programming languages the set of legal programs is recursive and it is only specialized features of languages such as those found in PL/1 which have enabled us to prove that the set of legal PL/1 programs is not recursive [Mandl 1969a].) This was the motivation for studying hierarchies of canonic systems. Doyle, in his Master's thesis, picked up this line of research and defined a partial hierarchy of canonic systems, trying to include in it correspondents for Chomsky's 4 types of formal grammars. Doyle's hierarchy has two distinct parts. The first part includes two classes of canonic systems, one equivalent in strong generative power to regular grammars and the other equivalent in strong generative power to context-free grammars:

> Theorem D-3 ["3" for "Type 3"]. The class of right-linear canonic systems and the class of regular grammars are strongly equivalent.
>
> Theorem D-2. The class of normal-form two-premise canonic systems and the class of context-free grammars are strongly equivalent.

There was a clear correspondence between the two formal systems, to each

production in the grammar corresponding a canon in the canonic system, and vice-versa. All the predicates occurring in the canonic system were of degree 1 (sets of strings), and the canonic systems turned out to be, in our terminology, simple canonic systems. In the second part of his hierarchy, Doyle allows predicates of degree 2 to occur (sets of pairs of strings) but no predicates of higher degrees, and obtains a class of canonic systems equipotent to Turing machines: for any grammar of Type 0 there is a canonic system which generates the same language. In other words,

Theorem D-0. The class of canonic systems with predicates of degree 2 is weakly equivalent to the class of Thue semisystems (grammars of Type 0).

From the proof of this theorem we also have:

Theorem D-0s. The class of simple canonic systems with predicates of degree 2 is weakly equivalent to the class of Thue semisystems.

Doyle also mentions "noncontracting canonic systems with predicates of degree 2", and states that these canonic systems generate only recursive sets and that for any given context-sensitive grammar one can find a "noncontracting canonic system with predicates of degree 2" weakly equivalent to it. We have not listed this as a theorem since the definition of "noncontracting" is entirely inadequate, especially when predicates of degrees 2 (and higher) are included, and therefore the above-mentioned class cannot be considered to be defined. In this connection, see also Chapter V.

This completes the second part of the hierarchy. The one-to-one

correspondence between the productions of the formal grammars and the canons of the corresponding canonic systems, while present in the first part of the hierarchy, could not be established in the second part, owing to the inherent difference between canons of these classes of canonic systems and the productions of T1 or T0 grammars. If we direct our attention to canonic systems which do take context into consideration (canonic systems with indicated context, which are here called 'general canonic systems'), a natural solution presents itself which not only fills in the above-mentioned gaps but actually brings about strong equivalences with all 4 types of formal grammars considered by Chomsky and with any type of grammar definable in terms of productions, thus embedding the theory of formal grammars into that of canonic systems. This simulation of formal grammars by appropriately restricted canonic systems with indicated context is the object of the present chapter.

The following definitions are analogous to Chomsky's:

Definition 2. A canonic system is called canonic system of Type 0 if each of its canons, except for five of them, is of one of the forms

(1) $x\varphi A\psi y$ derivable $\vdash$ $x\varphi\omega\psi y$ derivable

(2) A nonterminal

(3) a terminal

where

(a) φ, ψ, ω denote particular strings, possibly empty;

(b) A is a nonterminal (i.e. there is a corresponding canon of the form (2)); and

(c) for every symbol from the alphabet there is either a canon of

form (2), or a canon of form (3) (but not both),

the five other canons being

(4) $\vdash \Sigma$ derivable $(\Sigma \in V)$

(5) $\vdash \lambda$ terminal string

(6) x terminal $\vdash$ x terminal string $(x, y \in M)$ *

(7) x terminal ; y terminal string $\vdash$ xy terminal string

(8) x derivable ; x terminal string $\vdash$ x sentence .

We may dispense with the predicate 'nonterminal' altogether, and replace the present requirement (c) with a new one, (c'):

(c') Any symbol in the position of A in a canon of form (1) ** must not appear in a canon of form (3) .

Since this modification will simplify the proof of the main equivalence theorem, we shall adopt it.

---

* The effect of applying Canon (6) in a derivation can be achieved by applying Canons (5) and (7). Canon (6) was retained in order to preserve the correctness of future references by formula number.

** There are two ways in which a canon like

xABCy derivable $\vdash$ xABACy derivable

may be interpreted as a canon of form (1):

1) φ = A ; ψ = C ; ω = BA ; the expanded letter is B

2) φ = AB ; ψ = λ ; ω = AC ; the expanded letter is C .

(Of course, this is just one canon, not two, and the two interpretations have no influence on the use of this canon in derivations.) In such a case, only one of the symbols that may be considered as being "the expanded letter" is requied to be a nonterminal (i.e. to be missing from the canons of form (3)).

Definition 3. A canonic system is called a canonic system of Type 1 or context-sensitive canonic system (CSCS) if it is a canonic system of type 0 satisfying the additional condition that in all its canons of the form (1) the string ω is non-null.

Definition 4. A canonic system is called a canonic system of Type 2 or context-free canonic system (CFCS) if it is a CSCS satisfying the additional condition that in all its canons of form (1) the strings φ,ψ are null.

Definition 5. A canonic system is called a canonic system of Type 3 or regular canonic system if it is a CFCS satisfying the additional condition that in all its canons of form (1) the string ω contains just two symbols, one terminal and one nonterminal (one for which there exists a canon of the form (3) and one for which there is no such canon), always in the same order. If the order is "nonterminal - terminal", the regular canonic system is also called a left-linear canonic system.

The definitions for linear, one-sided linear, metalinear, sequential, etc., grammars can be similarly imitated, and so we can speak (Definition 6) of linear, one-sided linear, metalinear, sequential, etc., canonic systems. Obviously, all results obtained for these types of grammars hold also for the corresponding types of canonic systems.

Theorem 1 For every Type i canonic system (i = 0, 1, 2, 3) there is a Type i grammar which generates the same language, and conversely. In other words, the class of Type i grammars is equivalent to the class of Type i canonic systems for i = 0, 1, 2, 3.

We shall show how one can pass from grammars to canonic systems and from canonic systems to grammars. Let there be given a grammar G = N, T, P, Σ) of Type i (i = 0, 1, 2, 3). The associated canonic system has the canons (4),

(5), (6), (7), (8), one canon of the form (3) for each element of T, and for each production $\varphi A \psi \rightarrow \varphi \omega \psi$ one canon of the form (1). The resulting canonic system is, by construction, a canonic system of Type i (i = 0, 1, 2, 3); the strings φ, ψ may be empty. Suppose now a canonic system of Type i is given; the corresponding grammar is defined in the following manner. The set T includes all symbols for which there is a canon of type (3); N will include all other symbols and for each production there will be a canon of form (1). It is obvious that the resulting grammar is by construction of the same type as the canonic system from which it was derived.

Before we show how derivations are simulated, we should clarify what is meant by a derivation in formal grammars. Two definitions are in use in the theory of formal grammars, and our construction below works with either of them. According to the first definition, any sequence of applications of productions constitutes a derivation of the string obtained at the last application; a string is accepted iff:

a) it has a derivation;

b) it contains only terminal symbols.

According to the second definition, a sequence of applications of productions constitute a derivation only if no further applications of productions are possible. The grammar is usually required to have for each nonterminal symbol, at least one production expanding it, in which case a derivation produces automatically a string of terminals (if there were a nonterminal in the string, the sequence could be continued and therefore does not constitute a derivation); a string is accepted iff it has a derivation. We shall use the first definition, but we remark that if the grammar is required to have for each nonterminal symbol at least one production

expanding it, a derivation in the first sense (according to the first definition) is also a derivation in the second sense (i.e. cannot be continued) iff its last string contains only terminal symbols, and so the two concepts of acceptance coincide.

Let us consider a derivation in the canonic system. We shall simulate the derivation in the canonic system, in a step-by-step manner, by a derivation in the formal grammar. Without loss of generality, we may assume that the derivation in the canonic system starts with the canon (4). The derivation in the formal grammar simulating it will start with the one-character string Σ. Any canon of the form (1) will be simulated by means of the corresponding production; canons of other forms will be disregarded for the moment. We have thus obtained a derivation in the formal grammar simulating step-by-step the given derivation in the canonic system. If, the last string obtained is not only derivable but also a sentence, then this string has been obtained by applications of canons (5), (6), (7), with a final application of canon (8). The applicability of canon (8) proves that the second condition for acceptance in formal grammars (condition 'b)' of the first definition of derivation) is fulfilled, and therefore the string is accepted by the formal grammar.

Therefore we have shown that for every derivation in the canonic system there is a derivation in the grammar. The converse result is proved similarly. This completes the proof. It is easily seen that what we have proved amounts to strong equivalence. We can therefore assert:

(strong form of the general equivalence theorem)

Theorem 1' The class of Type i grammars is strongly equivalent to the class of Type i canonic systems, for i = 0, 1, 2, 3. The classes of linear, one-sided linear, metalinear, sequential, etc. grammars are strongly equivalent, respectively, to the classes of linear, one-sided linear, metalinear, sequential, etc. canonic systems.

Special types of context-sensitive canonic systems

Definition 7 A canonic system is called a left-context-sensitive canonic system (lCSCS) if it is context-sensitive and in all its canons of type (1) the right context (the string ψ) is empty. [And similarly for rCSCS.]

These definitions are the natural counterparts of the definitions for left-context-sensitive, [right-context-sensitive] grammars. One-sided context-sensitive grammars have been studied but with no significant results to date. About all that is known is that they can generate non-CF languages (and cannot generate non-context-sensitive languages). It is conjectured that they cannot generate all context-sensitive languages.

Another type of formal languages (left-context-sensitive) have been defined in Mandl 1968 and shown to be weakly equivalent in generative power to context-sensitive grammars. This gives rise to a new type of canonic systems strongly equivalent to left-context-sensitive grammars. These grammars seem to be new and interesting and therefore we will discuss these further here.

The definition below was suggested by Booth's definition of context-sensitive grammars [Booth 1967] as a phrase-structure grammar all of whose productions of any of the following three forms:

(9) $\zeta_1 \; A \; \zeta_2 \rightarrow \zeta_1 \quad \zeta_2 \quad \omega$

(10) $\zeta_1 \; A \; \zeta_2 \rightarrow \omega \quad \zeta_1 \quad \zeta_2$

(11) $\zeta_1 \; A \; \zeta_2 \rightarrow \zeta_1 \quad \omega \quad \zeta_2$

He further remarks that productions of the forms (9) and (10) are not really necessary (since they can be obtained by adding a few rules of the form (11) and by adding a few new nonterminal symbols) but they make his exposition easier to follow. Suppose now that the right contexts are null in all these rules (and similarly for left contexts). Then the rules have the form

ζ_1 A → ζ_1 ω

ζ_1 A → ω ζ_1

ζ_1 A → ζ_1 ω

where the first and the third are left-context-sensitive rules and the second is not. This second form of production will be the only form allowed in the grammars we are going to define.

Definition 8. A left-*context-sensitive grammar is a phrase-structure grammar all of whose productions except perhaps for a rule $\Sigma \rightarrow \lambda$, are noncontracting productions of the form

(12) $\varphi A \rightarrow \varphi\omega$, $A \in N$, $\varphi \in V^* = (N \cup T)^*$, $\omega \neq \lambda$.

[Similarly for right-*context-sensitive grammars] It may be remarked that this type of production is not a particular case of the general production $\varphi A \psi \rightarrow \varphi\omega\psi$ as the left-context-sensitive rules were. Likewise, the corresponding type of canon is not a particular case of (1), and so we cannot (yet) define left-*context-sensitive systems as a special case of T1 (or Type 0, for that matter) canonic systems (see footnote). We shall use instead a definition which is similar to Definition 11.

Definition 9. A left-*,context-sensitive, canonic system is a canonic system which includes the particular canons (4), (5), (6), (7), (8), a finite number of canons of the form

(14) $x\varphi Ay$ derivable $\vdash$ $x\omega\varphi y$ derivable

and one canon of form (3) for each symbol A occurring in some canon (14). [Similarly for right-*,context-sensitive, canonic systems; (14) is replaced by (15) $xA\psi y$ derivable $\vdash$ $x\psi\omega y$ derivable .]

Theorem 2. For any given context-sensitive grammar, there exists a left-*context-sensitive grammar (a right-*context-sensitive...) generating the same language and obtainable from the original one by a uniformly effective procedure. (The converse result is trivial.)

Proof. The proof will make use of certain reductions [Kuroda 1964] but it will be evident how to start the proof should one wish not to use the reductions. Definition [Kuroda] A context-sensitive grammar is of order n if there appears no string of length greater than n in any rule of the grammar. Lemma 1 [Kuroda] For any context-sensitive grammar of order n (n > 3) there exists a context-sensitive grammar of order n-1 generating the same language.

(By repeated use of Lemma 1:)

Lemma 2. [Kuroda] For any context-sensitive grammar there is a grammar of order 2 equivalent to it.

Let G be the given grammar. By introducing new terminal symbols, we can convert it to an equivalent grammar in which terminal symbols appear only in rules of the form A → a ("terminal rules").

Remark. [Kuroda] The original grammar might have been given in an apparently more general form* in which there might be a production which rewrites more than one symbol:

$$(16) \qquad \omega_1 \rightarrow \omega_2 \qquad \begin{pmatrix} |\omega_1| & \leq & |\omega_2| \\ \omega_1 & \in & (T \cup N)^* \cdot N(T \cup N)^* \end{pmatrix} .$$

---

*We can thus define two new types of canonic systems ("Types 1' and 0'"), with canons (4), (5), (6), (7), (8), canons of the form (3) (and (2)) and canons of the form

$$(17) \qquad x\omega_1 y \ \underline{\text{derivable}} \ \vdash \ x\omega_2 y \ \underline{\text{derivable}}$$

where ω_1 includes at least one nonterminal, with or without the restriction $|\omega_1| \leq |\omega_2|$. Using Kuroda's remark and our general equivalence theorem, we can conclude that these types of canonic systems are weakly equivalent, respectively, to T1 and T0 canonic systems. At this stage we could redefine left-*context-sensitive canonic systems as a certain special case of T1' (context-sensitive) canonic systems.

Using the general equivalence theorem, we have:

Theorem 3a. For any given left-*context-sensitive grammar there is a left-*context-sensitive canonic system strongly equivalent to it, and conversely. The class of left-*context-sensitive grammars is strongly equivalent to the class of left-*context-sensitive canonic systems.

Theorem 3b. For any given context-sensitive grammar (canonic system) there is a left-*context-sensitive canonic system (grammar) which generates the same language, and conversely. The class of context-sensitive grammars (canonic systems) is weakly equivalent to the class of left-*context-sensitive canonic systems (grammars).

Similar theorems hold for right-*context-sensitive canonic systems and grammars. A further application of the general equivalence theorem yields:

Theorem 4. For any given context-sensitive canonic system there is a left-*context-sensitive canonic system weakly equivalent to it. (The converse is trivial.)

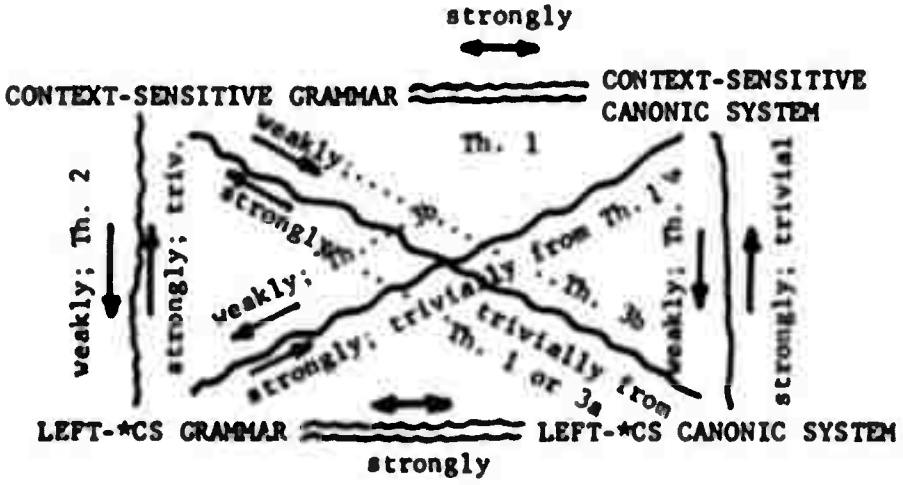


Figure 3

Most of the equivalence theorems of this chapter are summarized in Figure 3. For completeness' sake, we also included several trivial results.

CHAPTER III

SUBRECURSIVE CLASSES OF CANONIC SYSTEMS

Both these hierarchies of canonic systems, as well as the hierarchy of formal grammars, have no class of system to correspond to the class of recursive sets. ("Noncontracting canonic systems* with predicates of degree 2" were claimed to be situated somewhere between context-sensitive sets and recursive sets, both inclusions being in the weak sense.)

We state here in what sense(s) would a class of canonic systems (formal grammars, etc.) correspond to recursive sets and elucidate why no class of system has been found equivalent to recursive sets.

It is well known that there can be no procedure for deciding whether an arbitrary recursively enumerable set is a member of a given non-empty collection of recursively enumerable sets, except in the trivial case when all the recursively enumerable sets are members of the collection. This is Rice's theorem; see, e.g., Rogers [1967 , p. 324 (Th. 14-XIV (a))]. Consequently, it is clear that we cannot hope to find a class of canonic systems which (a) defines all recursive sets, and only recursive sets, and (b) the class includes all the canonic systems which define recursive sets.

---

* [Mandl 1969b]

We might hope that there exists a "small" class of canonic systems which define all and only recursive sets however realizing that the class cannot include all canonic systems which define recursive sets. Or, stated in another way, it might be the case that a certain class of canonic systems (characterized by a finite set of properties, and such that is is decidable whether a given canonic system meets those properties), would correspond to the recursive sets in the sense that

(a)
- only recursive sets are generated by canonic systems of that class (the class is "subrecursive");
- for every recursive set, there is among the canonic systems of that class at least one canonic system defining the given recursive set (and there may be such canonic systems outside the considered class).

We shall prove that such a class cannot exist, i.e. if a class of canonic systems defines only recursive sets, then it cannot define all recursive sets, even if it does not have a monopoly in defining recursive sets. This result can be restated succinctly as: "Subrecursive classes of canonic systems are strictly subrecursive."

Theorem 5. No class of canonic systems (or of any finitely-specified formal systems, for that matter) can correspond exactly (in the sense of (a) above) to the class of recursive sets. In particular, [NCST] ⊊ [Recursive sets] *.

*The reader may have noticed a similar statement, without proof, in Donovan and Doyle, 1968, p. 461 "Thus, a noncontracting canonic system can only define a recursive set. However, it cannot define all recursive sets; some recursive sets can be generated only to a TO grammar.". An earlier work claimed to have proved this by exhibiting a concrete example, but the proof was eliminated when the example turned out to be a context-sensitive set.

Proof (based on an idea of Hopcroft and Ullman [1969, §8.3]).

Since canonic systems are finitely specified, we can canonically enumerate all canonic systems, the canonical index encoding the whole description of the canonic system ("Gödelization" of canonic systems.) Likewise, we can canonically number (encode) all the words over the denumerably infinite list of potential symbols; let ω_k be the kth work in this numbering. Since it is assumed decidable whether a certain canonic system is of this type or not, we can strike out all the canonic systems not_of_this_type, thereby effectively enumerating all the canonic systems of the type considered: $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \ldots$. By the hypothesis, all these canonic systems define recursive languages $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3 \ldots$. Consider the set

$$\{\omega_k \mid \omega_k \notin \mathcal{L}_k\}$$

It is different from all $\mathcal{L}_i$, $i = 1, 2, \ldots$; yet it is recursive. Therefore no type of canonic systems can define all and only recursive sets.

Remark. A recursion-theoretic argument yields Theorem 7 as an immediate consequence of the known theorem that the class (set) of all recursive sets [while recursively enumerable as a class of r.e. sets [Blum 1965; Suzuki 1959]] is not characteristically enumerable. Proof_of_the_reduction. For all subrecursive classes of canonic systems the proof of the subrecursiveness has been done by exhibiting a decision procedure. In other words, if we have a finite description of a canonic system, we can interpret it not only as giving a procedure for enumerating a set but also as giving

a procedure for computing the characteristic function of the set, i.e. that we can find not only an r.e. index of the generated set but also a characteristic index thereof. Therefore [a description for] a canonic system belonging to a subrecursive class is akin to a characteristic index for the recursive set defined by that canonic system*.

---

*The elucidation of this point owes much to a discussion with Professor Patrick Fischer and Professor Juris Hartmanis at the Third Princeton Conference on Information Sciences and Systems in March 1969.

CHAPTER IV

CANONIC SYSTEMS FOR CONTEXT-SENSITIVE SETS

In Chapter II we mentioned Doyle's work on a hierarchy of canonic systems, where, inter alia, it was stated that the NCST were situated somewhere between context-sensitive sets and recursive sets. Let us now take a closer look at the definition of NCST. It reads ("Definition 2.13"):

> "A noncontracting canonic system (NCCS) is a canonic system in which each application of a canon results in the lengthening of the string denoted by the predicate defined in the canon. That is, if $\underline{A} \in P$ and $\omega \in \underline{A}$ and to prove $\omega \in \underline{A}$ it was first necessary to prove $\beta \in \underline{B}$, then $|\omega| \geq |\beta|$. That is, in a derivation, if we have
>
> $$\ldots ; \beta \ \underline{B} \ ; \ \ldots \ ; \ \omega \ \underline{A} \ ; \ \ldots$$
>
> then $|\omega| \geq |\beta|$. ($\underline{B}$ may denote the same predicate as $\underline{A}$)
>
> A noncontracting canonic system with predicates of degree two (NCST) can be constructed to describe the language generated by a T1 grammar; this canonic system has the same basic structure as the canonic system equivalent ot a T0 grammar with the additional length restriction."

Objections to the definition

1. "the string denoted by the predicate defined in the canon" . The conclusion of a canon has only one statement , and therefore it involves exactly one predicate. However, this predicate is not necessarily of degree 1 , so we cannot refer to "the string".

2. "lengthening" . That unspecified string is longer than something. Longer than what? The hypothesis of a canon may include many strings and many n-tuples (tuples) of strings.

3. " $|\omega| \geq |\beta|$ " . If ω and β are tuples, their length is undefined. If they are strings, then something has to be said about tuples, or at least about pairs, since predicates of degree 2 have to be allowed in order for Doyle's proof of [Type 1] $\subseteq$ [NCST] to work.

4. [Concerning the derivation] Although on p. 18 of that paper it was said "In this paper, a derivation will consist of a sequence of canons instead of the sequence of conclusions of these canons", here we have to revert to the original definition of derivation (as sequence of conclusions). When we do so, we see that an axiom may appear anywhere in this sequence, and it is not necessarily longer than all its predecessors (or shorter than other strings that may follow). Moreover, not only strings appear in a derivation but also tuples.

5. "(B may denote the same predicate as A)" . B does not denote a predicate; rather, it is a predicate. Formally, predicates are and remain elements of P ; and when we write $P = \{A , B\}$ we also mean that A and B are different elements of P . We could have introduced meta-variables ranging on predicates, $\mathcal{V}_1 , \mathcal{V}_2 , \ldots$, in much the same way in which we tacitly introduced $\beta , \omega , \varphi , \psi$ to stand for particular strings, and in that case we could have written

$$\ldots ; \beta \mathcal{V}_1 ; \ldots ; \omega \mathcal{V}_2 ; \ldots$$

and said that the meta-variables $\mathcal{V}_1$ and $\mathcal{V}_2$ may denote either two distinct predicates A , B or one and the same predicate A . Since we have not introduced such "predicate-variables", and since A , by definition, is not the same as B , one should have said

"... if we have ... ; β B ; ... ; ω A ; ...
or we have ... ; β A ; ... ; ω A ; ...
then $|\omega| \geq |\beta|$."

We therefore see that, at this stage, there is no such thing as non-contracting canonic systems with predicates of degree 2 . Correspondingly, this chapter will be devoted not to proving something about the [undefined] NCST but rather to finding a definition which will be intuitively acceptable and will be such that

1] Doyle's claims will hold for it ([Type 1] $\subseteq$ [new class] $\subseteq$ [Rec]);

2] will allow a proof of equivalence with [Type 1] .

As it very often happens in such cases, the real problem is not to prove but to "guess" what to prove (and to "improve a bad guess" by trial and error).

We cannot define 'noncontracting' [nc] as "such that the sum of the lengths of all strings in the hypothesis (whether appearing isolated or as elements of tuples) is at most as large as the sum of the lengths of all the strings in the conclusion" , since then a canon like

$$x\ \underline{A}\ ;\quad x\ \underline{B} \vdash x\ \underline{C}$$

would not be noncontracting, which is not only counter-intuitive but also does not allow us to salvage the proof for "[Type 1] $\subseteq$ [NCST] ". For the particular case when no predicates of degree 2 appear in the conclusions of the canons, one could try

> "the string in the conclusion is no shorter than any of the strings appearing in the hypothesis, whether they constitute terms of degree 1 or are elements of higher-degree terms" .

We shall reconsider this suggestion later on (in a modified form); at the moment we have to abandon it because we plan to use as much as possible of the existing proof [Donovan & Doyle 1968, pp. 43-44], and the canonic systems constructed in this proof are, as we noted in Chapter II, general canonic systems with predicates of degree 2 (also in the conclusions of the canons).

Since the real problem here was the finding of of a good definition, we think it would be more instructive for the student of canonic systems if we try to present how the definition was arrived at, instead of just exhibiting it and showing that it works.

Doyle's proof of the recursiveness used a multitape Turing machine; the idea was to show that this machine always halts, thus deciding membership in $L(\zeta)$. We intend to prove more, viz. that the set $L(\zeta)$

defined by the canonic system is context-sensitive. For this, it will be enough to show that the multitape Turing machine which decides whether $\omega \in L(\mathcal{C})$ never uses more than $|\omega|$ squares on any of its tapes.

As our first step, we modify Doyle's Turing machine to have, in addition to one tape for each predicate of degree 1, also k tapes for each predicate of degree k, for k = 2, 3, ... (all the tapes are distinct). In Doyle's construction, the Turing machine exhaustively generated all strings of length $\leq |\omega|$ in the language defined by the canonic system and checked for the occurrence of ω on the tape assigned to the sentence predicate. Naturally, all strings, on all tapes, had to be placed one beside the other (separated by special characters), and so the storage space for far from being linear. One could achieve linearity if each string replaces the contents of the tape on which it is placed, instead of being appended (with a separator) to the current end of the tape. However, each string has to stay available indefinitely, for later use in derivations (Fig. 4).

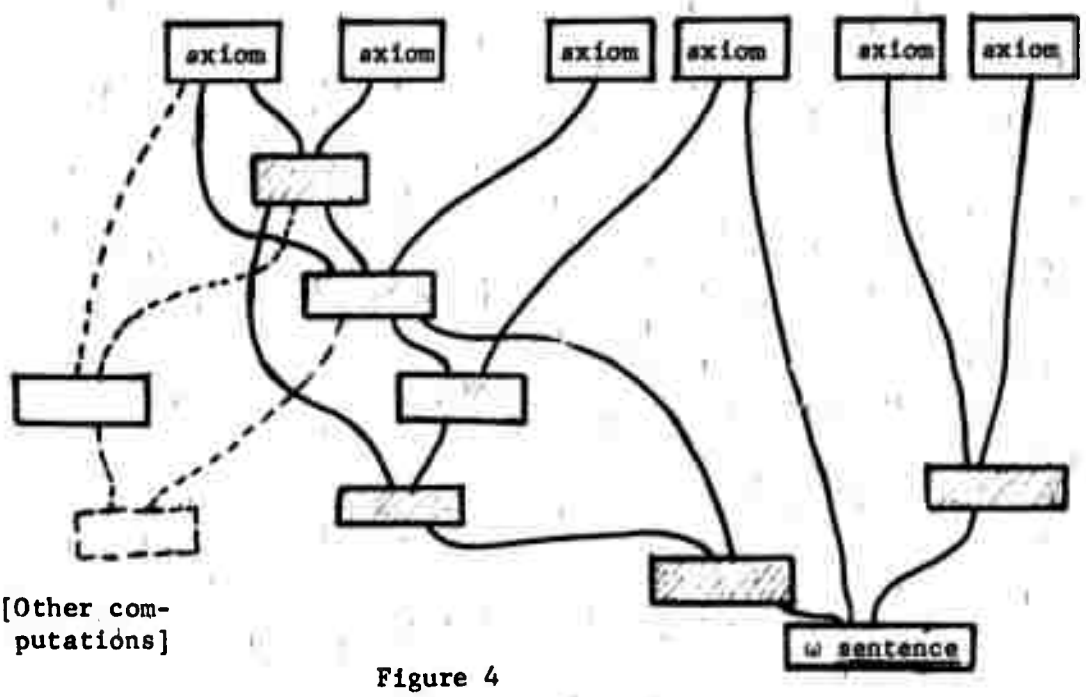


Figure 4

More exactly, it has to stay indefinitely available in all cases EXCEPT when each canon has at most one premise (if 0 premises, the canon is an axiom), in which case (see Fig. 5),

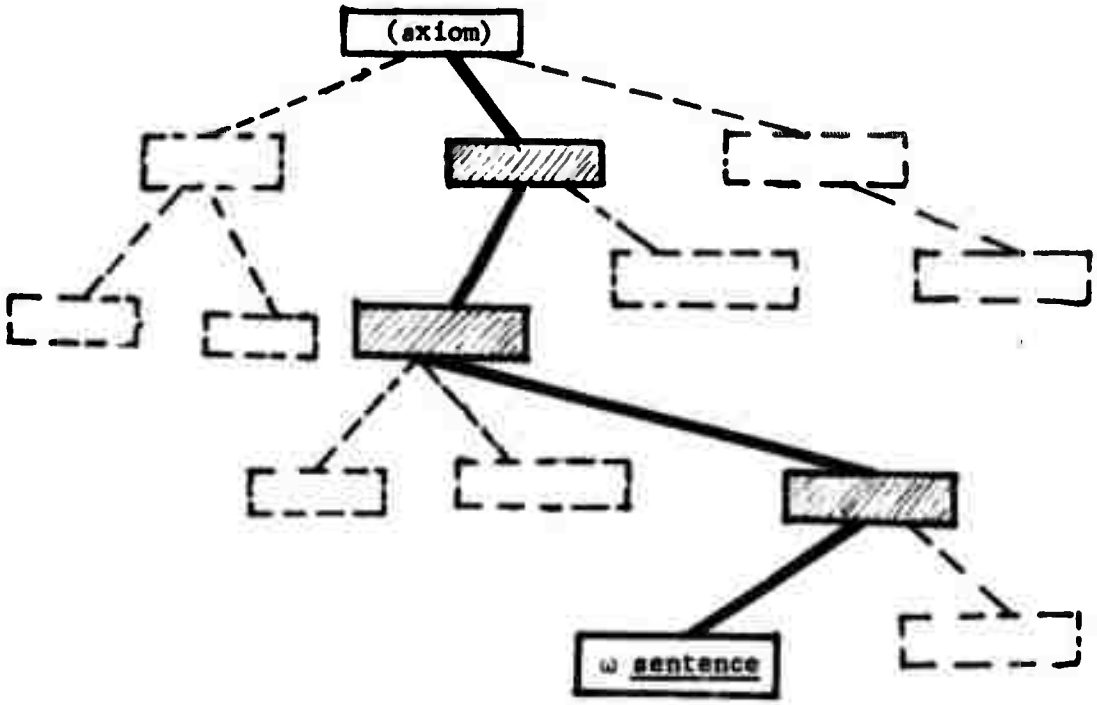


Figure 5

each statement on a computation path is used once immediately after being obtained and never needed again. This will be the main idea of our proof.

In order to achieve this situation we have to reduce our given canonic system $\mathcal{C}$ ("of Type X") to one $\mathcal{C}_N$ in which canons have at most one premise and which is also "of Type X" . Forgetting for the moment of the "Type X" restriction, we notice that such a reduction is always possible: this is one of Haggerty's results (Theorem H-3, here). There are exactly 3 ways in which the canons of $\mathcal{C}_N$ are constructed:

1) they may be inherited from $\mathcal{C}$, if they have at most one premise;

2) they may have been included in $\mathfrak{G}_N$ to replace some canon

$$t_1 \underline{pred}_1 \; ; \ldots ; \; t_n \underline{pred}_n \vdash t_o \underline{pred}_o$$

of $\mathfrak{G}$; general form:

$$\langle t_1 , t_2 , \ldots , t_n \rangle \; \underline{pred}_1 \underline{pred}_2 \ldots \underline{pred}_n \vdash t_o \underline{pred}_o \quad ,$$

where the degree of the newly introduced predicate is the sum of the degrees of the n predicates in the hypothesis of the old canon;

3) they may have been required by canons already in $\mathfrak{G}_N$:

if $\mathfrak{G}_N$ has canons

$$\langle t_1 , \ldots , t_m \rangle \; \underline{R} \vdash t_o \underline{R'}$$
$$\langle t'_1 , \ldots , t'_n \rangle \; \underline{S} \vdash t'_o \underline{S'} \quad , \text{ and } \underline{R'S'} \text{ is already in } \mathfrak{G}_N ,$$

then it will also have the canon

$$\langle t_1 , \ldots , t_m , t'_1 , \ldots , t'_n \rangle \; \underline{RS} \vdash \langle t_o , t'_o \rangle \; \underline{R'S'} \quad ,$$

where $\deg(\underline{RS}) = \deg(\underline{R}) + \deg(\underline{S})$, $\deg(\underline{R'S'}) = \deg(\underline{R'}) + \deg(\underline{S'})$.

From here we get the final hint as to how to choose "Property X" : if we are to use the method of proof sketched above, "Property X" has to be invariated by '2), '3)' . '3)' suggests the following:

PROPERTY X_1 . In each canon of the canonic system:
If the predicate in the conclusion is of degree k , then in each premise, separately, the tuple can be decomposed * into k parts (possibly empty), which are contiguous, mutually disjoint, and collectively exhaustive; and there is a permutation of these k parts such that, for every i , $1 \leq i \leq k$, each element in the ith part ** always represents a string which is no longer than that represented by the ith element of the term [of order k] in the conclusion of the canon.

---

* It is understood that no element of any tuple is to be cut in the middle by the decomposition.

** The part which became the ith after the application of the permutation.

Examples: $\langle x, y\rangle \underline{A} \vdash \langle x2, y3\rangle \underline{AA}$

$\langle x, y\rangle \underline{B} \vdash \langle y3, x2\rangle \underline{BB}$

As a particular case, we have:

PROPERTY X_2 . [Same as X_1 , but only one of the elements in the conclusion is compared with those in the hypothesis· there is an integer m , $m \leqslant k$, such that the mth element of the conclusion always represents a string longer than those represented by any element in any term in the hypothesis.]

Example: $\langle x, y\rangle \underline{A} \vdash \langle xy, 5\rangle \underline{B}$

We shall now clarify what we mean by the expression 'always represents a shorter string' . When a canon is used in a derivation it does not appear in its general form but as a particular canon instance, in which all the variables are replaced by particular strings. What Properties X_i ($i = 1 , 2$) require is that for each canon there be a decomposition of the kind specified above and such that for all the instances of that canon that can appear in derivations in the given canonic system * the above-mentioned decomposition yield particular strings which satisfy the length relationships specified in the definition.

---

* For example, if a canonic system contains only the canons

$\vdash$ 3 digit

$\vdash$ 5 digit

x digit $\vdash$ x number

x digit ; y number $\vdash$ xy number

then '535 digit $\vdash$ 535 number' is a legitimate instance of one of the above canons, but can never appear in a derivation. We shall be concerned here with canons like

$\langle x, y\rangle$ greater in length ; y very long string $\vdash$ x very long string

which are so decomposable, because any [apparently] offending instance

Thus in order to ascertain whether a certain c.s* has Property X_1 we have to make sure not only that the canons have certain forms but also that an infinity of canon instances satisfy certain restrictions. When we talk of classes of canonic systems we usually require that membership in the class be determined on the basis of a finite set of canons, not on the basis of an infinite set of canon instances; therefore we now proceed to define properties similar to Properties X_1 but such that they involve the canons themselves rather than an infinity of canon instances.

Let us consider first a term of degree 1 , e.g. xaby , where a , b are symbols and x , y are variables. Whatever the strings represented by x , y may be, the resulting string is always longer than the string represented by xxyabb . We shall write:

$$yx \prec xaby \prec xxyabb$$

Other examples:

$$x \prec xx$$

$$x \prec xy$$

$$x \prec xb$$

$$xx3y \succ yx \quad .$$

We have to make one more preparatory digression before we formally define the relation $\prec$. Since we want to use Doyle's construction of a c.s. for a given context-sensitive grammar, let us have a closer look at that construction. (We want to make sure that the definition of $\prec$ will be chosen in such a way that the c.s. constructed will have Property X_1 .) Its "most im-

---

<abc , defg> greater in length ; defg very long string ⊢ abc very long string , while legitimate as an instance, can never appear in a derivation in a c.s. which defines '<x , y> greater in length' to mean " x is longer than y ".

* "c.s." = "canonic system" .

portant canon", and the only one which is likely to cause problems, is

(1) wxz derived string ; ⟨x , y⟩ production ; ⟨y , x⟩ greater in length ⊢ wyz derived string .

The problem is that we need $wxz \leqslant wyz$, where x , y are not comparable (being two distinct variables). All we want is that always the string represented by y be at least as long as that represented by x , and this is ensured by the premise ⟨y , x⟩ greater in lenth ($|y| \geq |x|$). The definition will include also this case, thus "legalizing" canon (1) . The predicate greater in length used above is defined thus:

(2) x terminal ⊢ x symbol

(3) x nonterminal ⊢ x symbol

(4) ⊢ ⟨λ , 1⟩ length

(5) ⟨x , y⟩ length ; z symbol ⊢ ⟨xz , y1⟩ length

(6) ⟨x , y⟩ length ; ⟨z , y1⟩ length ⊢ ⟨z , x⟩ greater in length *

(7) ⟨x , y⟩ greater in length ; ⟨y , z⟩ greater in length ⊢ ⟨x , z⟩ greater in length

(8) ⟨x , y⟩ length ; ⟨z , y⟩ length ⊢ ⟨x , z⟩ greater in length

Canonic systems which include the canons (2)...(8) will be called length-monitoring canonic systems. We remark for later use that these canons satisfy themselves the requirements placed upon canons of canonic systems satisfying Properties X_1, X_2 (i.e. they are decomposable in the prescribed manner).

---

* It is because of this canon that the c.s. which include canons (2)... (8) are not simple. The second element of a pair in length represents the length of the first element expressed in 1-ary notation: 0='1', 3='1111', etc.

Definition 10. (Definition of $\preccurlyeq$ (with respect to a particular canon in a particular canonic system))

.1a. For any words α , β

$\lambda \preccurlyeq \alpha$ (λ is the empty word)

$\alpha \preccurlyeq \beta$.$\leftrightarrow$. $|\alpha| \leq |\beta|$

.1b. If x is a variable, then

$\lambda \preccurlyeq x$

$x \preccurlyeq x$

.1c. If a premise of the form $\langle v, u \rangle$ greater in length is included in the canon, where u , v are variables, then

$u \preccurlyeq v$ (in that canon)

If t_1 , t_2 , t_3 , t_4 represent concatenations of variables and words,

.2a. [Transitivity] $t_1 \preccurlyeq t_2 \cdot t_2 \preccurlyeq t_3 \cdot \Rightarrow t_1 \preccurlyeq t_3$

.2b. [Side-by-side concatenation of inequalities]

$t_1 \preccurlyeq t_2 \cdot t_3 \preccurlyeq t_4 \cdot \Rightarrow t_1 t_3 \preccurlyeq t_2 t_4$

.3. No relationship $t_1 \preccurlyeq t_2$ is valid unless it is deduced from a finite number of instances of .1a. , .1b. , .1c. by means of a finite number of applications of .2a. , .2b. .

With the help of the relation $\preccurlyeq$ we are now in a position to define PROPERTIES Y_1 , Y_2 , for length-monitoring canonic systems. These properties are defined in a similar manner to that in which we defined Properties X_1 , X_2 , but:

1) the expression 'element t_1 always represents a string which is no longer than that represented by t_2 ' is replaced by ' $t_1 \preccurlyeq t_2$ ' ;

2) the canons (2)...(8) , present in any length-monitoring c.s.,

are not required to be "decomposable" . [Notice the formal change in the concept of "decomposability".] [We shall later consider other types of length-monitoring c.s., in which case '2)' will refer to the canons there used for monitoring length.]

We note that Property Y_i implies Property X_i ($i = 1, 2$), and that one can immediately tell, by inspection, whether a c.s. has Property Y_i ($i = 1, 2$) or not (this was not the case for Property X_1 , Property X_2). This latter fact justifies the following definition:

Definition 11. A length-monitoring canonic system is of Type Y_1 (respectively Y_2) if it has the Property Y_1 (respectively Y_2) . [The name 'type' is reserved for properties detectable by inspection.]

Theorem 6.

a) Given any context-sensitive grammar, one can uniformly effectively construct a length-monitoring canonic system of Type Y_1 (Y_2) defining the same language.

b) For any length-monitoring canonic system of Type Y_1 (Y_2) , the language defined by it is context-sensitive (and a suitable grammar can be constructed in a uniformly effective manner).

Proof. Since Type Y_2 implies Type Y_1 , it is enough to prove 'a)' for Y_2 and 'b)' for Y_1 :

$$[\text{Type 1}] \underset{a}{\subsetneq} [\text{Type } Y_2] \subseteq [\text{Type } Y_1] \underset{b}{\subsetneq} [\text{Type 1}]$$

["the class of languages for which there is grammar of Type 1 is included in the class of languages defined by c.s. of Type Y_2 , which ..!", etc.] .

a) All we have to show is that the length-monitoring c.s. constructed in Donovan & Doyle 1968 pp. 43-44 always satisfies Property Y_2 , and this

is ensured by the manner in which we chose our definitions.

[Remarks. There is no need to first reduce the grammar to one of order 2;

- the alphabet of the c.s. includes not only the terminals but also the nonterminals, and Σ is included among the latter;
- there is no need for the canonic system itself to define the concept 'string', since this concept is part of the definition of canonic systems in general;
- for formal reasons, the canon

  [y string ;] $\langle \Sigma, y \rangle$ production ; $\langle y, \Sigma \rangle$ greater in length $\vdash$ y derived string

  is replaced by the two canons $\vdash \Sigma$ initial string and x initial string ; $\langle x, y \rangle$ production ; $\langle y, x \rangle$ greater in length $\vdash$ y derived string ,

  where initial string is a new, singleton predicate.]

b) Applying Theorem H-3 * , we reduce the given c.s. of Type Y_1 to one in which no canon has more than one premise. Since the original c.s. had Property X_1 , and since this property is invariated by the construction in Theorem H-3 , the resulting c.s. also has Property X_1 . We shall now construct (in a uniformly effective way) a nondeterministic multitape LBA which recognizes the language defined by the reduced c.s. (which is the same as that defined by the original one). For each predicate of degree k (k = 1 , 2 , ...), the LBA will have k tapes. Since each hypothesis has only one canon, the derivations have a certain "Markovian" character (see Fig. 5). Each statement obtained in the derivation is used in the immediately following step and never needed again, and therefore can allow ourselves to overwrite

* I am grateful to Amitava Bagchi for the suggestion to use Theorem H-3 in this proof.

the tapes corresponding to a predicate when this predicate reappears in a derivation. The LBA will simulate nondeterministically the derivation and will halt when a sentence is derived; if a string ω is a sentence then there is a computation path of the LBA which halts with ω displayed on the sentence tape, and conversely. The last step in the derivation of ω is of the form

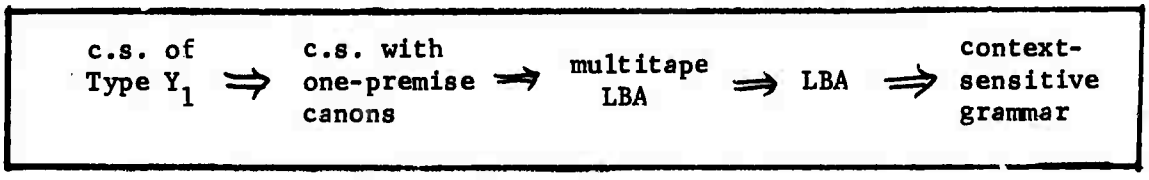
$$< \alpha_{\blacktriangleleft} \beta_{\blacktriangleleft} \cdots_{\blacktriangleleft} \mu > \underline{AB...M} \vdash \omega \ \underline{\text{sentence}} \quad ;$$

by Property X_1 we have

$$\begin{aligned} |\omega| &\geqslant |\alpha| \\ |\omega| &\geqslant |\beta| \\ &\vdots \\ |\omega| &\geqslant |\mu| \quad . \end{aligned}$$

Tracing back our derivation, we see that, in view of the Property X_1, ω is at least as long as any string in the derivation, and therefore $|\omega|$ is an upper bound, on each tape separately, on the amount of space necessary for recognition.

The proof will now be concluded by replacing the multitape LBA by a ["multitrack"] one-tape LBA and noting that each step in the chain of constructions



is uniformly effective.

As an illustration to this proof, we now show how the multitape LBA would handle the canonic system which was chosen by Haggerty to illustrate his procedure.

< x , y , z >	ABC	⊢ < ax , by , cz >	ABC
< y , z >	BC	⊢ < a , by , cz >	ABC
< x , y >	AB	⊢ < ax , by , c >	ABC
< x , z >	AC	⊢ < ax , b , cz >	ABC
x	A	⊢ < ax , b , c >	ABC
y	B	⊢ < a , by , c >	ABC
z	C	⊢ < a , b , cz >	ABC

< x , y >	BC	⊢ < xb , yc >	BC
x	B	⊢ < xb , c >	BC
y	C	⊢ < b , yc >	BC
		⊢ < b , c >	BC

< x , y >	AC	⊢ < xa , yc >	AC
x	A	⊢ < xa , c >	AC
y	C	⊢ < a , yc >	AC
		⊢ < a , c >	AC

Derivation for 'aabbbcaabbb' : b B ; < a bb > AB ; < aa bbb > AB ; < c aabbb > CD ; aabbbcaabbb E ;

The multitape LBA has 16 tapes (=5·1+4·2+1·3) . The following figure (Figure 6) shows the contents of these tapes at successive stages of the simulated derivation.

Original canonic system:

$$\vdash a\ \underline{A}$$
$$\vdash b\ \underline{B}$$
$$\vdash c\ \underline{C}$$
$$x\ \underline{A} \vdash ax\ \underline{A}$$
$$x\ \underline{B} \vdash bx\ \underline{B}$$
$$x\ \underline{C} \vdash cx\ \underline{C}$$
$$x\ A\ ;\ y\ \underline{B} \vdash xy\ \underline{D}$$
$$x\ \underline{C}\ ;\ y\ \underline{D} \vdash yxy\ \underline{E}$$

Derivation for 'aabbbcaabbb' : b $\underline{B}$; a $\underline{A}$; bb $\underline{B}$; aa $\underline{A}$; bbb $\underline{B}$; c $\underline{C}$; aabbb $\underline{D}$; aabbbcaabbb $\underline{E}$;

Transformed canonic system:

$$\vdash a\ \underline{A}$$
$$\vdash b\ \underline{B}$$
$$\vdash c\ \underline{C}$$
$$x\ \underline{A} \vdash ax\ \underline{A}$$
$$x\ \underline{B} \vdash bx\ \underline{B}$$
$$x\ \underline{C} \vdash cx\ \underline{C}$$
$$\langle x_{<}\ y \rangle \underline{AB} \vdash xy\ \underline{D}$$
$$\langle x_{<}\ y \rangle \underline{CD} \vdash yxy\ \underline{E}$$

(The decompositions are shown by suitable underlining)

$$\langle x_{<}\ y \rangle\ \underline{AB} \vdash \langle ax_{<}\ by \rangle\ \underline{AB}$$
$$x\ \underline{A} \vdash \langle ax_{<}\ b \rangle\ \underline{AB}$$
$$y\ \underline{B} \vdash \langle a_{<}\ by \rangle\ \underline{AB}$$
$$\vdash \langle a_{<}\ b \rangle\ \underline{AB}$$

$$\langle x_{<}\ y_{<}\ z \rangle\ \underline{ABC} \vdash \langle cz_{<}\ xy \rangle\ \underline{CD}$$
$$\langle x_{<}\ y \rangle\ \underline{AB} \vdash \langle c_{<}\ xy \rangle\ \underline{CD}$$

(2) (1) (1) (2)

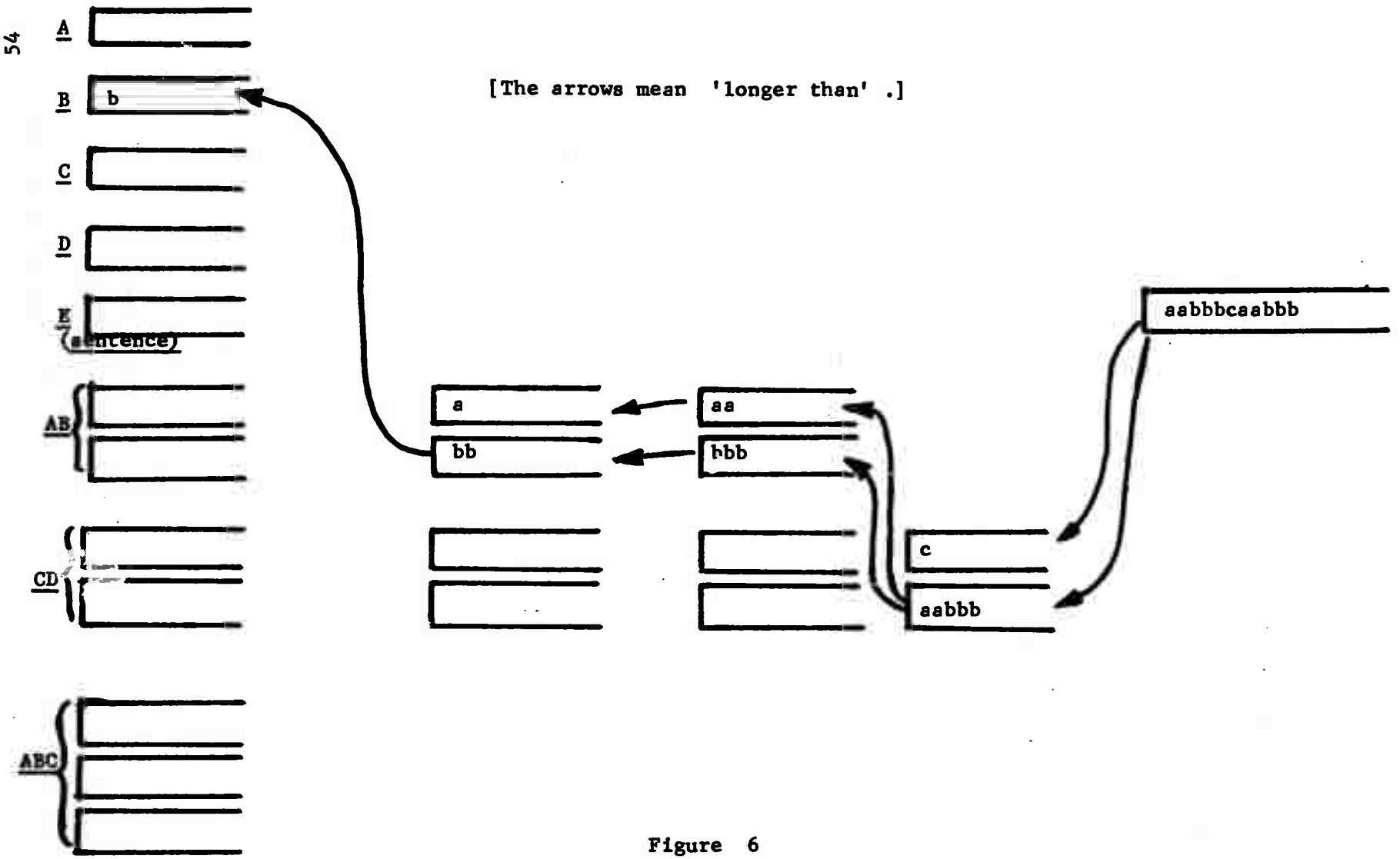


Figure 6

A 16-tape LBA simulates a derivation in a canonic system

BC (not used)

AC (not used)

CHAPTER V

FURTHER HIERARCHIES OF CANONIC SYSTEMS

The purpose of this chapter is to apply the main result of Chapter IV toward the development of improved hierarchies of canonic systems.

Let us consider Doyle's hierarchy again. This hierarchy has two separate parts, one part comprising classes of canonic systems strongly equivalent to the class of regular grammars and the class of context-free grammars, and the other part comprising a class of canonic systems weakly equivalent to the class of unrestricted rewriting systems (Thue semisystems). The hierarchy was claimed to include another class of canonic systems, situated somewhere between context-sensitive grammars and recursive sets, but we have seen in Chapter IV that this class was not completely defined. In the same chapter, two classes of canonic systems, the length-monitoring canonic systems of Type Y_1 (Y_2), were proved to be weakly equivalent to the class of context-sensitive languages. Therefore if we add any of them to the two parts of Doyle's hierarchy we obtain a complete hierarchy of canonic systems, where by "complete" we mean only that all 4 types of grammars are represented. (The hierarchy presented in Chapter II had correspondents not only for the 4 classic types of formal grammars but also for any class definable in terms of productions.)

While completeness is certainly a very desirable property, we cannot consider ourselves satisfied with it and ignore the fact that this combined hierarchy is quite heterogenous: for Types 3 and 2 it provides

simple canonic systems with predicates of degree 1 ; for Type 0 - simple canonic systems with predicates of degree 2 ; and for Type 1 the canonic systems are not even simple. The form of the hierarchy may be schematically summarized as

	S1	S1	G2	S2
(for Types:	3	2	1	0)

Our first step toward "homogenization" will be to reduce the third class from G2 to S2 . Clearly, we can always reduce a general c.s. to a simple one by using Theorem H-2 , but we need a class of simple c.s., weakly equivalent to context-sensitive grammars, and the property 'obtainable from class $\mathcal{A}$ by eliminating contextual references' is not a good criterion for class membership, since a criterion should refer to the form of the new system, irrespective of how the c.s. was obtained. We have seen that the length-monitoring c.s. cannot be simple, by definition, since they all include the offending canon

$$\langle x, y\rangle \text{ length} ; \langle z, yl\rangle \text{ length} \vdash \langle z, x\rangle \text{ greater in length}$$

If we modify IV.(2)...(8) by replacing this canon by the canons

(1) $\begin{cases} \langle x, y\rangle \text{ length} ; \langle z, yu\rangle \text{ length}; u \text{ unit} \vdash \langle z, x\rangle \text{ greater in length} \\ \vdash 1 \text{ unit} \qquad \text{[singleton predicate]} \end{cases}$

and call the canonic systems which include (1) and IV.(2)...(5),(7)...(8) s-length-monitoring canonic systems, we can build for them a theory similar to that of Chapter IV.

Definition 12. A simple s-monitoring canonic system is of Type Y_1 (respectively Y_2) if it has Property Y_1 (Y_2). Property Y_1 (Y_2)

for s-length-monitoring canonic systems is defined in a similar manner as for length-monitoring canonic systems, but the condition '2)' in that definition will now exempt from the decomposability requirement the modified canons used here for monitoring length.

Theorem 7.

a) Given any context-sensitive grammar, one can uniformly effectively construct a simple s-length-monitoring canonic system of Type Y_1 (Y_2) defining the same language.

b) For any simple s-length-monitoring canonic system of Type Y_1 (Y_2), the language defined by it is context-sensitive (and one can uniformly effectively find a grammar for it).

Proof. a) The only contextual referencing in the canonic systems of Theorem 6a was in Canon IV.(6) . If we replace that canon by (1) we get a canonic system which is simple, s-length-monitoring, of Type Y_2 (and therefore also Y_1) , and defines the same language.

b) Completely similar to the proof of Theorem 6b . [Theorem 7b is not a particular case of Theorem 6b since s-length-monitoring c.s. are, formally, not the same as length-monitoring c.s.]

We have thus obtained a hierarchy of the form

S1 S1 S2 S2 ,

i.e. a hierarchy of simple canonic systems (of which the last class contains all the simple c.s. with oredicates of degree 2), and we shall try to reduce it to the form

S1 S1 S1 S1 .

The last class can easily be so reduced. For any r.e. set there is a simple c.s. with predicates of degree 2 which defines the given set, and this c.s. may be reduced to one with predicates of degree 1 (by Theorem H-1) while remaining simple; and the converse result is certainly true, since sets defined by canonic systems are always recursively enumerable.

The hierarchy has now the form

S1 S1 S2 S1 .

Unfortunately, Theorem H-1 appears to be of no further use in reducing the form of the hierarchy, since none of the 4 classes mentioned in this chapter as being weakly equivalent to context-sensitive grammars (length-monitoring canonic systems of Type Y_1 ; of Type Y_2 ; simple s-length-monitoring c.s. of Type Y_1 ; of Type Y_2) is invariant under the transformation involved in the proof of Theorem H-1.

Having thus arrived at an apparent "dead end" in our endeavors to develop and simplify Doyle's hierarchy, we now consider the other basic hierarchy, the hierarchy of general c.s. with predicates of degree 1 (of the form G1 G1 G1 G1) which was introduced in Chapter II, and apply to it Theorem H-2. It is easily seen that we obtain indeed 4 types of canonic systems, i.e. valid criteria can be stated (depending only on the form of the transformed canonic system) for membership of a c.s. in a type. These types of c.s. may also be introduced independently. The following definitions are analogous to Definitions 2...5 .

Definition 13. A simple canonic system is of Type $0^{(s)}$ if each of its canons, except for 4 of them, is of one of the forms

(2) xuy derivable ; u U ; v V $\vdash$ xvy derivable

(3) $\vdash \mu$ M (x, y, u, v are variables)

(4) $\vdash$ s terminal

where

(a) μ (a meta-variable) stands for a particular string;

(b) for any predicate appearing in a canon of form (2), except for the canon derivable, there is exactly one canon of form (3), i.e. U, V, M are singleton predicates;

(c) if U, V (in this order) are two singleton predicates appearing in a canon of form (2), and if μ, ν are the corresponding strings, then μ and ν can jointly be put in the form

$$\mu = \varphi A \psi$$
$$\nu = \varphi \omega \psi$$

where φ, Ψ, ω are [meta-variables standing for] particular strings, possibly empty, and A does not appear in a canon of form (4),

the 4 other canons being:

(5) $\vdash \Sigma$ derivable

(6) $\vdash \lambda$ terminal string

(7) x terminal ; y terminal string $\vdash$ xy terminal string

(8) x derivable ; x terminal string $\vdash$ x sentence .

Definition 14. A simple canonic system is of Type $1^{(s)}$ if it is of Type $0^{(s)}$ and satisfies the additional condition that for each canon of form (2) the corresponding string ω (defined in (c)) is non-null.

Definition 15. A simple canonic system is of Type $2^{(s)}$ if it is of Type $1^{(s)}$ and satisfies the additional condition that for each canon of form (2) the corresponding strings φ, ψ (defined in (c)) are null.

Definition 16. A simple canonic system is of Type $3^{(s)}$ if it is of Type $2^{(s)}$ and satisfies the additional condition that for each canon of form (2) the corresponding string ω contains just two symbols, one terminal and one nonterminal (one for which there is a canon of form (4) and one for which there is no such canon), always in the same order.

The definitions of linear, one-sided linear, metalinear, sequential, left-, context-sensitive, etc., grammars may be similarly imitated, and so we may speak (Definition 17) of linear, one-sided linear, metalinear, sequential, left-, context-sensitive, etc., simple canonic systems.

Theorem 8 [Analogous to Theorem 1] . For any simple canonic system of Type $i^{(s)}$ ($i = 0, 1, 2, 3$) there is a grammar of Type i which generates the same language, and conversely.

Proof. Similar to that of Theorem 1.

The second part of Theorem 8 (the converse result) can be proved

more easily if we use Theorem 1 and the following obvious Lemma:

Lemma. The result of applying the procedure of Theorem H-2 upon a canonic system of Type i (i = 0, 1, 2, 3) is a simple canonic system of Type $i^{(s)}$.

Theorem 8 provides us with a hierarchy of simple canonic systems with predicates of degree 1, that is a hierarchy of the form

S1 S1 S1 S1 ,

and the goal of the present chapter is thereby completely achieved. Before concluding this chpter, however, we should like to point out an interesting fact which provides a link between the two basic hierarchies developed in this chapter (the one of the form S1 S1 G2 S2 -- based on Doyle's -- and the other of the form G1 G1 G1 G1 , introduced in Chapter II). When we wanted to reduce the first basic hierarchy to one composed exclusively of simple canonic systems and noticed that its third class, the length-monitoring c.s. of Type Y_1 , failed to be simple only because one of the canons used in monitoring string lengths included contextual referencing, we just replaced the offending canon. But there is absolutely no need for a canonic system to monitor itself the lengths of the strings. A context-sensitive grammar does not monitor the lengths of its strings, and it is no less noncontracting because of this; strings grow in length not because the grammar monitors their lengths (which it does not) but just because the productions are noncontracting. When we examine the grammar "from the outside" (by using a meta-system) we can prove that the strings are bound to grow; but there is no need to duplicate this proof inside the object system (the

grammar - or the canonic system). We therefore eliminate the canons IV.(2)...(8), and the canonic system becomes now simple. It still contains canons of the form

$$\vdash \langle \varphi A \psi \lessdot \varphi \omega \psi \rangle \text{ production}$$

(one for each production $\varphi A \psi \rightarrow \varphi \omega \psi$; Donovan & Doyle 1968, p. 43), and we just know that in each such canon $|\omega| \geq 1$. This, however, does not yet solve our problem. We have to redefine the concept 'c.s. of Type Y_1' , or, more exactly, to redefine the relation $\lessdot$; and this relation has to hold, sometimes, between two different variables, as for example, in

(9) wxz derived string ;$\langle x \lessdot y \rangle$ production; $\langle y \lessdot x \rangle$ greater in length $\vdash$ wyz derived string

where we ought to be able to prove that $x \lessdot y$. For length-monitoring c.s. we could say that $x \lessdot y$ because the premise $\langle y \lessdot x \rangle$ greater in length is present (Definition 10.1c), but we do not have the predicate greater in length any more, and we are still under the obligation to ascertain, by merely inspecting the canons, whether or not the canonic system if a member of the class we define. One way to solve the problem of eliminating length-monitoring and still being able to define $\lessdot$ is to abolish the need to ever compare (in length) two distinct variables. Then $\lessdot$ would become an absolute relation, not dependent on the canonic system, and defined by .1a.1b.2a.2b.3. of Definition 10 (i.e. without .1c.). To achieve this end we have to replace canon (9) by as many canons as there are productions, each new canon being the result of "plugging in" a particular production in the canon (9) :

(10) $w\varphi A\psi z$ derived string $\vdash$ $w\varphi\omega\psi z$ derived string

The class of canonic systems of Type Y_1 (from the first basic hierarchy) is thereby transformed into a class which is, essentially, no different from the class of canonic systems of Type 1 (from the second basic hierarchy; Definition 3), and from here the whole second basic hierarchy is just one small step away.

CHAPTER VI

NON-CROSSREFERENCING, SIMPLE, AND NON INSERTING CANONIC SYSTEMS

CLASSIFICATION OF CANONIC SYSTEMS

In this chapter we pursue an idea mentioned in Chapter I -- that one should not distinguish (and name) the subclass of canonic systems with contextual referencing, with insertions, with crossreferencing, but one should rather consider the subclasses of canonic systems without the respective options. Canonic systems without contextual referencing (simple c.s.) were extensively studied in Chapters I and V; we shall now formally introduce the other two classes and investigate their computational power.

Crossreferencing was defined [Donovan & Doyle 1968, p. 27] as consisting of the use of one and the same variable more than once in the term of the conclusion or the use of one and the same variable in more than one premise in the hypothesis. The possibility of a variable being used in exactly one premise of the hypothesis but occurring several times in that premise is not included in this definition. On the other hand, there is a fundamental difference between multiple occurrences in the hypothesis part of the canon and multiple occurrences in the conclusion. The applicability of a canon in a particular situation has to be established before the canon could be used, and the applicability depends only on the hypothesis of the canon; if the hypothesis contains two occurrences of a variable, we have to check that the strings matched by

the two occurrences are identical strings (substrings), and this checking is not an elementary action. Multiple occurrences in the conclusion, however, have no influence on the applicability of the canon. This argument suggests that we should specifically exclude from the definition of crossreferencing multiple occurrences in the conclusion, and include multiple occurrences of a variable within one premise of the hypothesis. The same point of view is taken by Turing (in connection with Turing machines) and by Minsky (in connection with Post's canonical systems). Quoting from Turing 1936 [p. 137 in Davis's collection]:

> "If, on the other hand, [the squares] are marked by a sequence of symbols, we cannot regard the process of recognition as a simple process. This is a fundamental point and should be illustrated. In most mathematical papers the equations and theorems are numbered. ... But if the paper was very long, we might reach Theorem 157767733443477; then, further on in the paper, we might find '... hence (applying Theorem 1577677334-3477) we have ...' . In order to make sure which was the relevant theorem we should have to compare the two numbers figure by figure, possibly ticking the figures off in pencil to make sure of their not being counted twice." .

Minsky [1967, p. 231] remarks that he could have allowed multiple occurrences of variables within any premise, but chose not to:

> "Post's most general formulation allowed each production to have several antecedents. ... Also in Post's most general formulation, he allowed two of the $'s in the antecedent to be the same. This meant that the rule of inference would apply only to a string (theorem) in which there was an exact repetition of some (variable) substring in two places in the antecedent. We prefer to prohibit antecedents of this form, not because we want to restrict the generality of the systems, but because it would run counter to our intuitive picture of what ought to be permitted as elementary, unitary operations." .

With this motivation (and backing) we change the definition of 'crossreferencing' to read:

Definition 18. A canon is said to contain crossreferencing if at least one of the variables involved in it occurs more than once in the hypothesis of the canon, whether these occurrences are within one premise or are in different premises.

Definition 19. A canonic system is non-crossreferencing if none of its canons contains crossreferencing.

Let us consider now the phenomenon of insertion, whose definition is implicit in the traditional definition of canonic systems with insertion as canonic systems in which terminal symbols are inserted between the variables of one string to form a new string. Since we are interested in canonic systems without insertions, we tentatively define canons without insertion as canons in whose conclusion no symbols appear, i.e. whose conclusions contain concatenations of variables rather than concatenations of variables and words. The formal modifications required in the defining second-level canonic system are not difficult to figure out, but the definition would be forbiddingly restrictive: the axioms would be totally useless. In fact, we never defined axioms formally, but just referred by this name to any canon whose list of premises was empty, and therefore any restriction on the canons is automatically a restriction on the axioms. This suggests the following definition:

Definition 20. A canonic system is non-inserting it it has the property that in all its canons, except for the axioms, the term in the conclusion of the canon has only "pure" elements, i.e. each element is either a concatenation of variables or a concatenation of symbols.

The following canonic systems will be used as examples:

1. Language: set of balanced (well-formed) strings of parentheses

$V = \{ (\ ,\ ) \}$ [Minsky p. 230]

$\vdash$ () theorem
x theorem $\vdash$ (x) theorem
x theorem $\vdash$ xx theorem
x()y theorem $\vdash$ xy theorem

one-predicate (Post)
non-crossreferencing

1'. Same language, same alphabet. [Minsky p. 230]

$\vdash$ () theorem
xy theorem $\vdash$ x()y theorem

simple
one-predicate
non-crossreferencing

2. Language: palindromes over a , b , c . [Minsky p. 228]

$\vdash$ a A
$\vdash$ b A
$\vdash$ c A
$\vdash$ aa A
$\vdash$ bb A
$\vdash$ cc A
x A $\vdash$ axa A
x A $\vdash$ bxb A
x A $\vdash$ cxc A

simple
one-predicate
non-crossreferencing

2'. Same language. [Minsky p. 228]

$\vdash$ a A
$\vdash$ b A
$\vdash$ c A
x A $\vdash$ axa A
x A $\vdash$ bxb A
x A $\vdash$ cxc A
x A $\vdash$ xx A

simple
one-predicate
non-crossreferencing

3. Language: all true statements about adding 1-ary positive integers [Minsky p. 229] (3 = '111' , etc.)

$V = \{1 , + , = \}$

$\vdash 1+1=1$ add

x+y=z add $\vdash$ x1+y=z1 add

x+y=z add $\vdash$ x+y1=z1 add

[or: x+y=z add $\vdash$ y+x=z add]

one-predicate
non-crossreferencing

inserting
not simple

4. Language: all true statements about multiplying 1-ary positive integers [Minsky p. 229]

$V = \{1 , \cdot , = \}$

$\vdash 1\cdot 1=1$ mult

$x\cdot y=z$ mult $\vdash x1\cdot y=zy$ mult

$x\cdot y=z$ mult $\vdash y\cdot x=z$ mult

one-predicate
non-crossreferencing

inserting
not simple

5. Language: $\{a^m b^n a^m b^n \mid m , n$ natural numbers $\}$

$\vdash a$ A

$\vdash b$ A

x A $\vdash$ ax A

x B $\vdash$ bx B

x A ; y B $\vdash$ xyxy sentence

simple
non-crossreferencing

inserting

5'. Same language.

$\vdash a$ A

$\vdash b$ B

x A ; y A $\vdash$ xy A

x B ; y B $\vdash$ xy B

x A ; y B $\vdash$ xyxy sentence

simple
non-inserting
non-crossreferencing

6. Language: squares in 1-ary.

Alphabet = { 1 , * }

⊢ 1* A

x*y A ⊢ x11*yx A

x*y A ⊢ y square

non-crossreferencing

not simple
inserting

6'. Same language.

[Mentioned here for completeness; will be introduced later.]

5''. Language: same as 5 .

⊢ a A

⊢ b B

x A ; y A ⊢ xy A

x B ; y B ⊢ xy B

simple
non-inserting

crossreferencing

x A ; y B ; z B ⊢ xyxz ABA

x A ; y A ; z B ⊢ xzyz BAB

x ABA ; x BAB ⊢ x sentence

5'''. Same language.

⊢ a A

x A ⊢ ax A

⊢ b B

x B ⊢ bx B

ABA
BAB as above

x ABA ; x BAB ⊢ x sentence

simple

inserting
crossreferencing

5''''. Language: $a^{m-1}b^n a^m b^n$ m , n natural numbers

The first 6 canons of 5''. non-inserting

Last canon replaced by

ax ABA ; x BAB x sentence not simple
crossreferencing

5^{v}. Same language.

The first 4 canons of 5' . non-inserting
non-crossreferencing

Last canon replaced by

x A ; y B xyxy ABAB

ax ABAB x sentence not simple

Of the classes of canonic systems considered until now, only the two classes which correspond, in the first basic hierarchy, to regular grammars and to context-free grammars are non-crossreferencing. Since they are also simple and non-inserting, this implies that non-inserting c.s., non-crossreferencing c.s., and simple non-crossreferencing non-inserting c.s. are all powerful enough to define any context-free language. The following figure shows the new classes of canonic systems (the numbers refer to our examples):

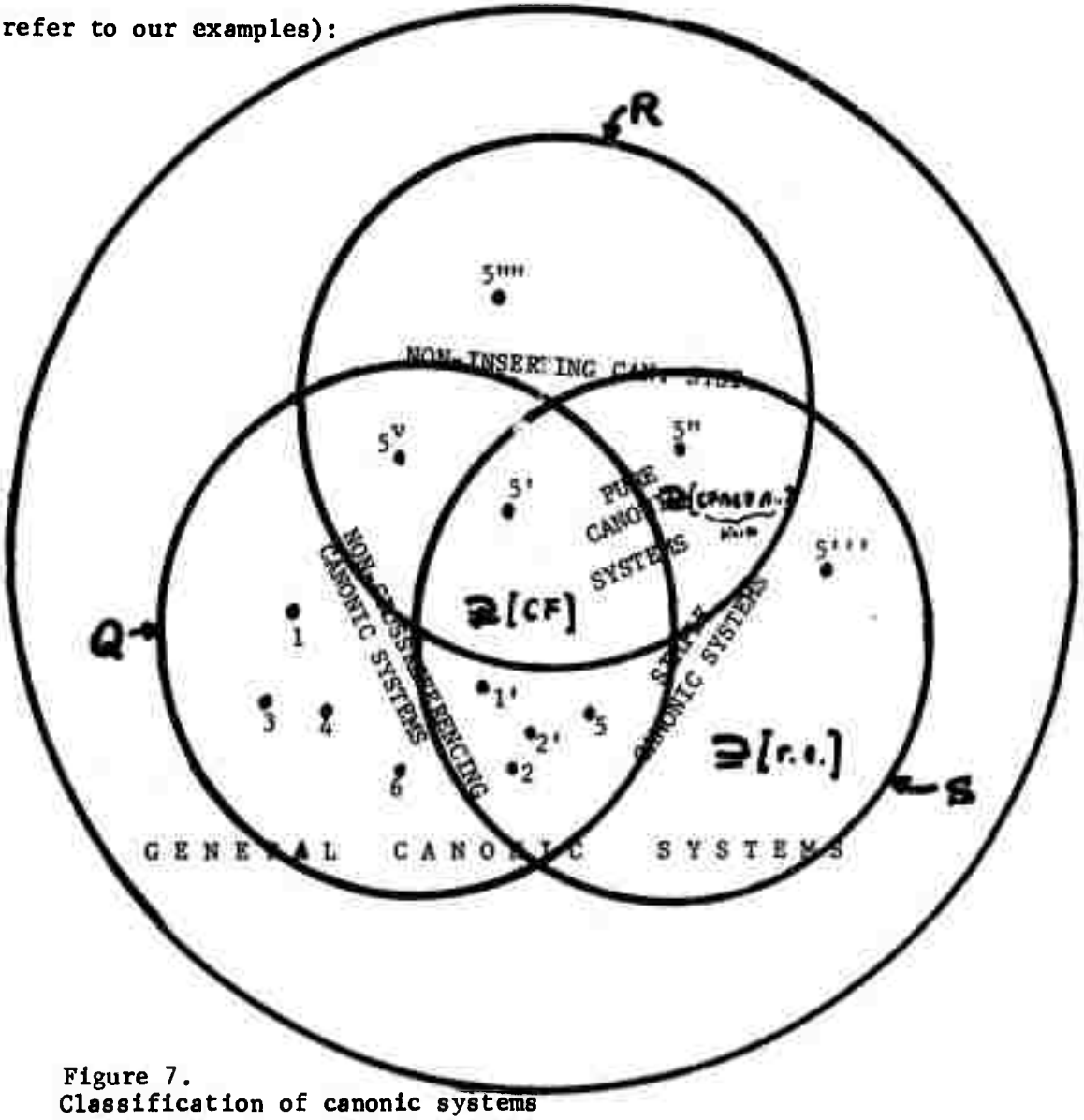


Figure 7.
Classification of canonic systems

Introducing the abbreviations

Q = non-crossreferencing c.s.

R = non-inserting c.s.

S = simple c.s.

$QR = Q \cap R$

$Q\bar{R} = Q \setminus R$ etc.

$QR\bar{S} = (Q \cap R) \setminus S$

we have

$[QRS] \supsetneq [CF]$ (since (5') defines a non-, context-free, language)

$\Rightarrow$ { $[QR] \supsetneq [CF]$

$[QS] \supsetneq [CF]$ As before, [class of c.s.] = class of languages definable by the class of c.s.

$[RS] \supseteq$ [finite intersections of CF languages]

(since they can be obtained by cross-referencing; we note that the language of (5') is included in this class)

$[Q] \supsetneq [CF]$

$[R] \supseteq$ [finite intersections of CF] [Result improved below]

$[S] = [r.e.]$.

Since a c.s. in QRS can be trivially modified so as to belong to $\bar{Q}RS$ or $Q\bar{R}S$ or $QR\bar{S}$, we also have

$[QR\bar{S}] \supsetneq [CF]$

$[Q\bar{R}S] \supsetneq [CF]$

Similarly,

$[\bar{Q}R\bar{S}] \supseteq$ [finite intersections of CF]

$[Q\bar{R}\bar{S}] \supsetneq [CF]$

We shall now show that the non-inserting c.s. are powerful enough to define any r.e. set: [R] = [r.e.] . As for the non-crossreferencing c.s., we have not been able to improve the result stated above ($[Q] \supsetneq [CF]$). It is known that non-crossreferencing c.s. with one predicate of order 1 cannot define the set of squares in 1-ary (see [Minsky 1967, p. 235]).

Theorem 9 [analogous to Theorem H-2 (of Haggerty)]. Any c.s. can be reduced to a non-inserting canonic system.

Proof. Any word (sequence of symbols) in the conclusion is replaced by a variable whose value is specified (by an additional premise) to be in an (adequately defined) singleton predicate. This implies that the desired reduction is possible.

Remark. This procedure invariates the class of non-crossreferencing canonic systems; it is recalled that the elimination of words from the hypothesis of a canon introduced crossreferencing.

We shall now develop a complete hierarchy of non-inserting c.s.. Since the first two classes from the first basic hierarchy (of the form S1 S1 G2 S2) are already non-inserting, we shall retain them and adapt the last two to our purposes.

The most general non-inserting c.s. obviously generates an r.e. set; and we have seen that for any r.e. set there is a non-inserting c.s. defining it (by Theorem 9). This gives us a class corresponding to Type 0.

As for Type 1 , we have a result completely similar to Theorem 7 (Ch. V). Before stating it, we need a few definitions. We would like to talk about non-inserting length-monitoring c.s.; but all length-monitoring c.s. (as previously defined) include the [inserting] canon

IV.(5) $\langle x, y\rangle$ length ; z symbol $\vdash \langle xz, yl\rangle$ length .

[A similar situation was encountered just before Definition 12.] By replacing this canon by

$\langle x, y\rangle$ length ; z symbol ; u unit $\vdash \langle xz, yu\rangle$ length
$\vdash$ 1 unit [singleton predicate]

in the definition of 'length-monitoring' we arrive at the definition of r-length monitoring c.s. (similar to s-length-monitoring c.s.). Similarly, if we perform this replacement in the definition of 's-length-monitoring' , we arrive at the definition of rs-length-monitoring canonic systems.

Theorem 10.

a) Given any context-sensitive grammar, one can uniformly effectively construct a non-inserting r-length-monitoring c.s. of Type Y_1 (Y_2) * defining the same language.

b) For any non-inserting r-length-monitoring c.s. of Type Y_1 (Y_2) * the language defined by it is context-sensitive (and one can uniformly effectively find a grammar for it).

---

* Definition similar to Def. 12.

Proof. Analogous to that of Theorem 7.

We have thus obtained a hierarchy of non-inserting canonic systems. The form of this hierarchy may be described as

R1 R1 R2 R2
↓
R1 .

(More exactly, RS1 RS1 RG2 RS1 .)

We shall now define pure canonic systems.

Definition 21. A canonic system is said to be pure if it is simple and non-inserting. ["pure" since all concatenations contain either exclusively symbols or exclusively variables.]

A complete hierarchy of pure c.s. , of the form

RS1 RS1 RS2 RS2
↓
RS1 ,

can be easily obtained, in a manner entirely similar to that in which the hierarchy of non-inserting c.s. was obtained. However, we prefer to present another hierarchy, based on the second basic hierarchy (form: G1 G1 G1 G1). At the end of Chapter IV, we found a hierarchy of simple c.s. with predicates of degree 1 : S1 S1 S1 S1 . (Cf. Def. 13, 14, 15, 16 and Theorem 8.) By inspecting the definitions of the classes of simple c.s. involved, it is easily seen that these canonic systems are also non-inserting. Therefore we have obtained:

Theorem 11. The hierarchy

S1 S1 S1 S1

of Theorem 8 is, actually, a hierarchy of pure canonic systems,

RS1 RS1 RS1 RS1 .

The next logical step would be to look for a hierarchy of the form QRS1 QRS1 QRS1 QRS1 . We suspect that such a result is impossible to obtain, and, more precisely, that the non-crossreferencing c.s. are not sufficiently powerful to define any language of Type 0 or 1 . We shall now introduce a modification in their definition, modification which will enable us to obtain a complete hierarchy. Following Minsky's [1967, p. 235] definition for Post systems, we shall call canonic system with auxiliary alphabet a formal system similar to ordinary c.s. but in which a subset T of the alphabet V is singled out (and called terminal alphabet), and for which the language it defines is

Definition 22.

$$T^* \cap \bigcup_{A \in D} L(\mathfrak{G}, A)$$

rather than $\bigcup_{A \in D} L(\mathfrak{G}, A)$. The set $V \setminus T$ is called auxiliary alphabet. Systems with $T = V$ may be identified with ordinary canonic systems.

The difference between canonic systems with auxiliary alphabet and ordinary canonic systems becomes significant only in the case of non-crossreferencing canonic systems. For all other canonic systems we could define a predicate terminal string and then achieve the desired effect by adding a canon like

x sentential form ; x terminal string $\vdash$ x sentence .

Example. The set of squares in 1-ary.

6'. $V = \{1, *\}$
$T = \{1\}$

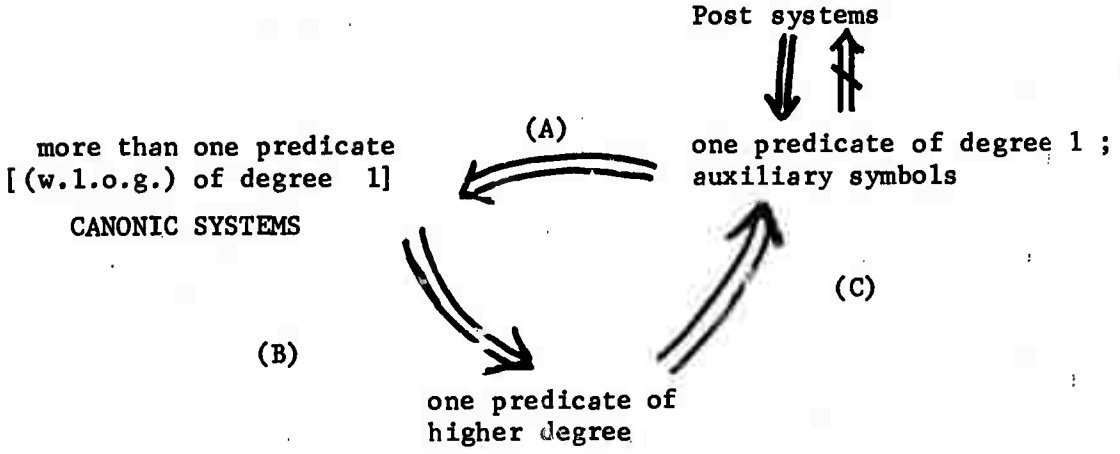
non-crossreferencing
one-predicate
WITH AUXIL. ALPHABET

Canons:

$$\vdash 1* \ A$$
$$x*y \ A \vdash x11*yx \ A$$
$$x*y \ A \vdash y \ A$$

not simple
inserting

Examples 6 and 6' show that a set may be undefinable by Post systems (canonic systems with one predicate of degree 1 and no auxiliary alphabet) but become definable if we either allow one more predicate or allow an auxiliary symbol. This "trade-off" between additional predicates and additional (auxiliary) symbols is, in fact, an instance of a general result:



(A): Trivial.

(B)(C) : [Haggerty 1969, p. 44] *

* Theorem 3. However, the statement of this theorem, "Any canonic system can be simulated by a Post system.", must be supplemented by the

(B) may also be proved by using Theorem H-3 and the proof of Theorem 6b. [The number of the predicates will be the number of tapes of the LBA.]

(C): Proved by introducing separators acting as auxiliary symbols.

A result similar to (B)(C) has been announced by N. Kohn [1969]; it involves variables which range on all but one of the symbols in the alphabet.

Having defined and exemplified canonic systems with auxiliary alphabet, we are now ready to derive a hierarchy of non-crossreferencing canonic systems with auxiliary alphabet.

Theorem 12.

a) Non-crossreferencing canonic systems with auxiliary alphabet are powerful enough to define any r.e. set.

b) A complete hierarchy of such canonic systems may be obtained from the second basic hierarchy.

Proof. Obviously, it is sufficient to prove 'b)' .

The only canon with crossreferencing in the canonic systems from the above-mentioned hierarchy (Chapter II) is

$$x \text{ derivable} \; ; \; x \text{ terminal string} \vdash x \text{ sentence} \quad .$$

By eliminating it from a given c.s. (together with the canons which define the predicate terminal string) and by replacing the axioms of the form

$$\vdash a \text{ terminal}$$

---

qualification "...which is a canonical extension [in Minsky's sense] of the given canonic system.", since there are formulas which are theorems in the Post system without being theorems in the given canonic system, and all such formulas contain auxiliary symbols not in the alphabet of the given canonic system. The canonic systems 6 and 6' above are examples of systems which can not be simulated unless we allow canonical extensions.

by a declaration

$$T = \{ a , \ldots \} ,$$

we obtain a canonic system equivalent to the given one. The theorem now follows.

OPEN PROBLEMS:

1. "[QRS] = ? " Find the computational power of the class of simple, non-crossreferencing, non-inserting canonic systems (no auxiliary alphabet, any number of predicates). [[QRS] $\supseteq$ [CF]]

2. " [Q] = ? " Find the computational power of the class of non-crossreferencing canonic systems (no auxiliary alphabet, any number of predicates). [Includes all finite intersections of context-free sets.]

3. " $[Q_1]$ = ? " Find the computational power of [unextended] Post systems (non-crossreferencing, no auxiliary alphabet, one predicate (necessarily of degree 1)).

REFERENCES

Alsop, Joseph W. 1967 A canonic translator , Project MAC Technical Report MAC-TR-46, M.I.T., Cambridge, Mass., June 1967.

Blum, E. K. 1965 Enumeration of recursive sets by Turing machine , Zeitschrift für mathematische Logik und Grundlagen der Mathematik 11, 197-201.

Booth, Taylor L. 1967 Sequential machines and automata theory , Wiley, New York.

Davis, Martin [Ed.] 1965 The undecidable - basic papers on undecidable propositions, unsolvable problems, and computable functions , Raven Press, Hawlett, N.Y.

Donovan, John J. 1966 Investigations in simulation and simulation languages , Doctoral Dissertation, Yale University.

Donovan, John J., and Doyle, James T. 1968 Hierarchies of canonic systems . Reissued in August 1969 as Project MAC Memorandum MAC-M-417.

Donovan, John J., and Ledgard, Henry F. 1967 A formal system for the specification of syntax and translation of computer languages , Proceed. F.J.C.C., 1967, pp. 553-569.

Doyle, James T. 1968 Issues of undecidability in canonic systems , S.M. Thesis, M.I.T., Cambridge, Mass., January 1968.

Doyle, James T. - see also Donovan, John J.

Haggerty, Joseph P. 1969 Complexity measures for language recognition by canonic systems , S.M. Thesis, M.I.T., Cambridge, Mass., January 1969.

Hopcroft, John E., and Ullman, Jeffrey D. 969 Formal languages and their relations to automata , Addison-Wesley.

Kohn, Norman 1969 The relationship between canonic systems and Post systems . Unpublished.

Kuroda, Sige-Yuki 1964 Classes of languages and linear-bounded automata , Inform. and Control 7, 207-223.

Ledgard, Henry F. - see Donovan, John J.

Mandl, Robert 1968 Topics in the theory of automata and formal languages . Term paper for Mathematical Models in Linguistics (M.I.T. 23.772), May 1968.

Mandl, Robert 1969a The place of PL/1 in the hierarchy of formal languages , Project MAC Memorandum MAC-M-419, January 1969.

Mandl, Robert 1969b Canonic systems and recursive sets , Proceed. of the Third Annual Princeton Conference on Information Sciences and Systems, p.363.

Minsky, Marvin 1967 Computation - finite and infinite machines , Prentice-Hall.

Rogers, Hartley, Jr. 1967 Theory of recursive functions and effective computability, McGraw-Hill.

Suzuki, Y. 1959 Enumeration of recursive sets , J. of Symbolic Logic 24, 311.

Turing, Alan M. 1936 On computable numbers, with an application to the Entscheidungsproblem , Proc. London Math. Soc. (2) 42 (1936-7), 230-65. Correction, ibid. 43 (1937), 544-6. Reprinted in Davis's collection, pp. 116-51 and 152-4.

Ullman, Jeffrey D. - see Hopcroft, John E.

LIST OF DEFINITIONS

			Page	Chapter
Def.	1	Simple canonic systems	8	I
	2	Canonic systems of Type 0	25	II
	3	" " " 1	27	
	4	" " " 2	27	
	5	" " " 3	27	
	6	Linear, sequential, etc., canonic systems	27	
	7	Left-CS canonic system	30	
	8	Left-*CS grammar	31	
	9	Left-*CS canonic system	31	
		Property X_1 , X_2	44	IV
		Length-monitoring canonic system	47	
	10		48	
		Properties Y_1 , Y_2	48	
	11	Types Y_1 , Y_2 for length-monitoring can. syst.	49	
	12	" " for simple s-" " "	56	V
	13	Simple canonic system of Type $0^{(s)}$	59	
	14	" " " " $1^{(s)}$	60	
	15	" " " " $2^{(s)}$	60	
	16	" " " " $3^{(s)}$	60	
	17	Linear, sequential, etc., simple can. syst.	60	
	18	Crossreferencing (64)	66	VI
	19	Non-crossreferencing canonic system	66	
	20	Non-inserting canonic system	66	
		Q, R, S and their combinations	72	
		Pure canonic system	75	
		Canonic system with auxiliary alphabet [a generalized canonic system, not a subclass of the ordinary canonic systems]	76	

LIST OF THEOREMS

		Page	Chapter
Th. 1	Equivalence theorem for Types 0, 1, 2, 3.	27	II
	G1 G1 G1 G1	27,58	
1'	General equivalence theorem (strong)	29	
2	Left-*CS ≈ context-sensitive	32	
3a	Equivalence between	33	
3b	context-sensitive and left-*CS	33	
4	grammars and canonic systems	33	
5	Subrecursive classes are strictly subrecursive	36	III
6	[Type Y_1] = [Type 1]	49	IV
	S1 S1 G2 S2	56	V
7	[Simple of Type Y_1] = [Type 1]	57	
8	Equivalence theorem for simple canonic systems of Types $0^{(s)}$ ... $3^{(s)}$	60	
	S1 S1 S2 S2	57	
	S1 S1 S2 S1	58	
	S1 S1 S1 S1	61	
	Connection between the two basic hierarchies	61	
9	Any c.s. can be reduced to a non-inserting c.s. ...	73	VI
10	[non-inserting ... Type Y_1] = [Type 1]	74	
	R1 R1 R2 R2	75	
	R1 R1 R2 R1	75	
	RS1 RS1 RS2 RS2 (hier. of pure c.s.)	75	
	RS1 RS1 RS2 RS1	75	
11	RS1 RS1 RS1 RS1	76	
12a	non-crossreferencing c.s. with auxil. alphabet can define sny r.e. set	78	
12b	hierarchy of non-crossreferencing c.s. with auxilisry alphabet	78	
	Theorems H-1, H-2, H-3 [Haggerty]	21	I
	Theorems D-3, D-2, D-0 [Doyle]	23	II

LIST OF FIGURES

		Page	Chapter
Fig. 1	Simple and general canonic systems	7	I
2	Parsing of a canon	11	
3	Relationships between left-*CS and CS grammar and c.s. ...	33	II
4	Derivations in general c.s.	42	IV
5	Derivations in c.s. in which canons have most one premise	43	
6	Multitape LBA simulating a derivation in a c.s. ...	54	
7	Classification of canonic systems	71	VI