The cumulative recurrence library

M. G. Notley*

* *Research and Advanced Development Organisation, International Computers Limited, Minerva Road, Park Royal, London NW10*

The Cumulative Recurrence Library (CRL) is an heuristic procedure for finding recurring strings of adjacent symbols within a text. The mechanism of the CRL is described in some detail. Some preliminary experimental results are quoted. Possible variations of the CRL procedure are discussed, and some applications of the CRL are considered.

(Received January 1969)

1. Introduction

The Cumulative Recurrence Library (CRL) is an heuristic procedure for finding recurring patterns of adjacent symbols within a text.

The symbols may be drawn from any finite alphabet such as, for example, digital control signals in a process controller (in which case the text consists of a sequence of control actions), or operation codes in a computer program.

The CRL was designed during a feasibility project when working on the application of heuristic techniques to the symbolic solution of equations. One approach considered involves a graph traverser (Michie, 1967) in which the nodes consist of partially solved sets of equations, and in which the development of a node consists of the application of one or more standard transformations to the equations to generate a new set of equations. It was decided that such a system could usefully be made adaptive by designing a procedure that allows the set of possible transformations that can be applied to be increased by the introduction of 'macro-transformations' which consist of commonly occurring sequences of standard transformations. In this manner it would be possible for the system to apply one macro-transformation instead of having to select each standard transformation in turn.

The procedure designed to achieve the generation of these macro-transformations is the CRL. The CRL is designed to be as general as possible, rote learning commonly recurring sequences of any symbols in any text under the most general parsing rules in the most efficient manner. For this reason it was felt that the CRL would be of wide general application.

Using the English language as an example, consider the problem of finding the most common words and phrases (sequences of symbols) using no *a priori* knowledge of the language. The most obvious approach would be to note each symbol as it occurred and set up frequency counts for all possible sequences of known symbols, *A*, *B*, *C*, . . . and *AA*,*AB*,*AC*, . . . *AAA*, *AAB*, *AAC*, . . . etc., up to some arbitrarily chosen maximum length sequence. It is evident that this system would be prohibitively costly in terms of time and computer storage space. Also (considering for example the frequency of such sequences as *QZX*), it would evidently be very wasteful. This is the problem that the CRL is designed to tackle.

2. The mechanism of the CRL

The CRL contains a set of strings of symbols, and with each string is associated a 'count'. A very simple example of the contents of a CRL is shown in **Fig. 1.** In this example the strings are shown numbered and arranged in order of decreasing count, also, for simplicity, only a very small number of strings have been shown and the library would normally contain many other strings such as *B*, *C*, *D*, *BA*, *AB*, *CB*, etc.

The strings contained in the CRL at any instant serve to parse an input text by selecting the longest completed string contained in the library that may be found at the front of the input text. **Fig. 2** illustrates an input text '*ABCAADCABCBA*' as it is parsed by the CRL shown in Fig. 1, where the strings are shown separated from the rest of the input text by brackets.

The object of the string counts is to control changes in the set of strings held in the library. Each time any string is parsed, all of the string counts are reduced by one and then the count of the string which actually occurred is increased by an amount *m*. It has been found by experiment that a suitable value for this increment *m* is the number of strings currently held in the library. (Thus in the example shown in Fig. 1 there are five strings, $m = 5$.)

NUMBER	STRING	COUNT
1	*AD*	9
2	*ABCB*	4
3	*BAC*	2
4	*ABC*	−6
5	*A*	−23

Fig. 1. Simple example of a CRL

The changes that occur in the string counts with this example, (as the first string, *ABC*, from the input is parsed), are shown in **Fig. 3.**

Alterations to the strings contained in the library may occur in two ways, by the formation of compound strings and by the formation of new single symbol strings.

ABCAADCABCB
(ABC)AADCABCBA
(ABC)(A)ADCABCBA
(ABC)(A)(AD)CABCBA

Fig. 2. Text parsed by CRL shown in Fig. 1

NUMBER	STRING	OLD COUNT	NEW COUNT
1	*AD*	9	8
2	*ABCB*	4	3
3	*BAC*	2	1
4	*ABC*	−6	−2
5	*A*	−23	−24

Fig. 3. Changes in count due to parsing the string *ABC*

Whenever the count for a particular string becomes greater than a threshold value then a new, compound, string is formed that consists of this string preceded by the string which, in the input text, was found to precede it. It will be noted that, since no string count will increase unless that string has been found in the text, any new string formed in this way must have occurred in the input text. A suitable value for the threshold has been found to be $2m$, where m is the number of strings currently held in the library. The counts for the new string just formed and the string whose count has exceeded the threshold are then set to the new value of m (the old m plus one).

The introduction of new single symbol strings into the library occur whenever the parsing system is unable to find any complete string that matches the first part of the input text. In this case the first symbol of the input text is introduced into the library as a single symbol string and given a count of zero. The CRL procedure then acts as if this new string had just been parsed, by subtracting one and adding the new m to the new string count.

In **Fig. 4** is illustrated a flowchart of the CRL procedure which may serve to summarise the mechanism of the CRL described so far.

Consider an initially empty library acting on a very simple language in which there are only two words '*CA*' and '*BA*' which occur equiprobably but randomly.

Fig. 5 illustrates the development of the counts of the contents of the CRL.

It will be seen that the contents of the CRL very quickly come to represent the features of the language. Over a longer period of time the counts of the strings '*A*' and '*B*' and '*C*' will get more and more negative and may, if required, be removed from the library, since none of these strings ever occur except as parts of '*CA*' or

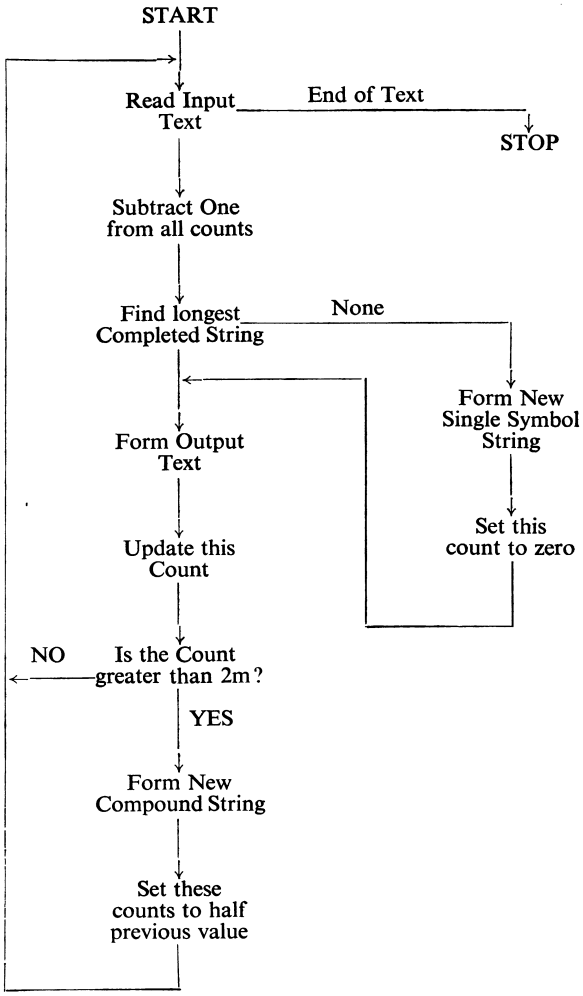

Fig. 4. Flowchart of the CRL procedure

'*BA*'. Also compound strings '*CABA*', '*BABA*' etc., may be formed but, since these will occur less frequently than the '*CA*' and '*BA*' strings, will tend to have lower counts. Thus it may be seen that the CRL will tend to generate, as strings with high counts, strings of symbols whose internal correlation is high. This would be reflected, for example, in the English language by common words and phrases.

3. Experimental results

A program has been written in FORTRAN for the KDF9 computer to test the CRL procedure. Two main experiments have been carried out using this program. In these experiments the English language was used since it is, perhaps, easier to appreciate the results obtained in such a familiar field. It must be noted that not just the alphabetic letters, but also the punctuation marks and also, which is more important, the wordspace or blank, were all treated as identically significant symbols so that no *a priori* knowledge of the language was assumed. (The interested reader should be able to hand simulate the program from the description of the CRL procedure given above in order to verify the results of the experiments.) The results quoted are copied directly from the computer output, the only changes being to rearrange the layout for ease of reading. The results consists of the way in which the CRL parsed the text that was

presented to it as the library developed, with the later parsing shown in any case in which compound strings were formed on two successive parsings. In each experiment the CRL started with an entirely empty library as in the example quoted in Fig. 5.

In the first experiment a short text for four simple and repetitive sentences was presented to the CRL five times in succession. The CRL started with an empty library on the first pass and then, on each successive pass, started with the library formed by the end of the previous pass. The results of this experiment are shown in **Fig. 6.** The speed with which the long compound strings are formed is high, as might be expected with repetitions of the same text, and the procedure is evidently efficient.

The second experiment involved a single pass of a long piece of text, rather than a series of passes of a short piece of text. The text used was a children's story (Eastman, 1962) which was chosen because it consisted of a text in Basic English with a maximum vocabulary of 100 words, (and also, perhaps, because it is a delightful story). The text was presented to the CRL broken up into a series of short sections which coincide with the printing on each separate page of the original book (accompanied in the book by a picture). The results of this experiment are shown in **Fig. 7.** (For those who are interested in the story, the SNORT is an enormous steam excavator.) It will be noted that the strings of symbols do not get as long as in the previous experiment but that they coincide much more closely with actual words and phrases (for example '*AND*', '*YOU ARE*', and '*MY MOTHER*' right at the end).

4. Comments on the CRL mechanism

The CRL is an heuristic procedure and it is evident that there are several possible ways of altering the parameters or detailed mechanism of the procedure to produce slightly different results. It is not possible completely to justify the exact procedure described above, but some discussion of the details chosen and of possible variations is required.

The method of parsing the input text used in the CRL procedure described above was chosen because it was believed to be the most generally applicable. It must be admitted that this belief is intuitive, but the method used has two great advantages; it is simple to implement, and it always results in an unambiguous parsed structure.

In the CRL procedure the counts associated with each string are obviously related to the frequency with which that string occurs. This relationship, however, is not simple. The actual count at any instant depends not only on the frequency but also on how recently that

Input text: *CACABACABACACABABACACABA*

1. *C*:	0	−1	0	−1	−2	−3	−1	−2	−3	−4	−2	−3	−1	−2	−3	−4	−5	−2	−3	0	−1	−2
2. *A*:	1	0	1	0	2	1	3	2	4	3	5	4	6	5	4	3	2	5	4	3	2	
3. *B*:	2	1	0	−1	1	0	−1	−2	−3	−4	−2	−3	−4	−5	−6	−7	−8	−9				
4. *BA*:	3	6	5	4	3	2	6															
5. *CA*:	4	3																				

Fig. 5. Development of empty library

Text: '*THE CAT SAT ON THE MAT. THE BLACK CAT SAT ON THE MAT. THE CAT SAT ON THE BLACK MAT. THE BLACK CAT SAT ON THE BLACK MAT.*'

Parsing:

First Pass

(T)(H)(E)( )(C)(A)(T)( )(S)(A)(T)( )(O)(N)( )(T)(H)(E)(M)(A)(T)(.)
( )(T)(H)(E)(B)(L)(A)(C)(K)( )(C)(A)(T)( )(SAT)(O)(N)( )(T)(H)(E)
(M)(AT)(.)( )(T)(H)(E)(C)(AT)( )(SAT)(O)(N T)(H)(E)(B)(L)(A)(C)
(K)( )(M)(AT)(.)( )(T)(HE)(B)(L)(A)(C)(K)( )(CAT)(SAT)(O)(N T)
(HE)(B)(L)(AC)(K)( )(M)(AT)(.)

Second Pass

(T)(HE)(CAT)(SAT)(O)(N THE)(M)(AT)(.)( )(T)(HE)(B)(L)(AC)(K)( )
(CAT)(SAT)(O)(N THE)(MAT)(.)(THE CAT)(SAT)(O)(N THE)(B)(L)
(AC)(K)( )(MAT)(.)(THE)(B)(LAC)(K)(CAT SAT)(ON THE B)(LAC)(K)
(MAT)(.)

Third Pass

(T)(HE)(CAT SAT)(ON THE MAT.)(THE)(BLACK CAT SAT)(ON THE MAT.)
(THE CAT)(SAT)(ON THE B)(LAC)(K)(MAT)(. THE)(BLACK CAT SAT
ON THE BLACK MAT.)

Fourth Pass

(T)(HE))(CAT SAT ON THE MAT.)(THE BLACK CAT SAT)(ON THE MAT.)
(THE CAT)(SAT)(ON THE B)(LAC)(K)(MAT)(. THE)(BLACK CAT SAT
ON THE BLACK MAT.)

Fifth Pass

(T)(HE)(CAT SAT ON THE MAT.)(THE BLACK CAT SAT ON THE MAT.)
(THE CAT SAT ON THE BLACK MAT. THE BLACK CAT SAT ON THE BLACK MAT.)

Fig. 6

Parsing of text from a children's story 'Are you my mother?' by P. D. Eastman. (100 Word vocabulary in Basic English.)

(*A*)(*R*)(*E*)( )(*Y*)(*O*)(*U*)( )(*M*)(*Y*)( )(*M*)(*O*)(*T*)(*H*)(*E*)(*R*)(.)

(*A*)( )(*M*)(*O*)(*T*)(*H*)(*E*)(*R*)( )(*B*)(*I*)(*R*)(*D*)( )(*S*)(*A*)(*T*)(*O*)(*N*)( )(*H*)(*E*)
(*R*)(*E*)(*G*)(*G*)(.)

(*T*)(*H*)(*E*)(*EG*)(*G*)(*J*)(*U*)(*M*)(*P*)(*E*)(*D*)(.)

(')(*O*)(*H*)( )(*OH*)(')(*S*)(*A*)(*I*)(*D*)( )(*T*)(*H*)(*E*)(*M*)(*O*)(*THE*)(*R*)(*B*)(*I*)
(*R*)(*D*)(.)( )(')(*M*)(*Y*)( )(*B*)(*A*)(*B*)(*Y*)(*W*)(*I*)(*L*)(*L*)( )(*B*)(*E*)(*H*)(*E*)
(*RE*)(.)(*H*)(*E*)(*W*)(*IL*)(*L*)(*W*)(*A*)(*N*)(*T*)(*T*)(*O*)(*E*)(*A*)(*T*)(.)(')

(')(*I*)(*M*)(*U*)(*S*)(*T*)(*G*)(*E*)(*T*)(*S*)(*O*)(*M*)(*E*)(*THI*)(*N*)(*G*)(*FO*)(*R*)(*M*)
(*Y*)(*B*)(*AB*)(*Y*)(*B*)(*I*)(*R*)(*D*)( )(*T*)(*O*)(*E*)(*AT*)(.)(')( )(*S*)(*H*)(*E*)
(*SAI*)(*D*)(.)(')(*I*)(*W*)(*IL*)(*L*)(*B*)(*E*)(*B*)(*A*)(*C*)(*K*)(.')

(*S*)(*O A*)(*W*)(*A*)(*Y S*)(*HE*)(*W*)(*E*)(*N*)(*T*.)

(*THE*)(*EG*)(*G*)(*J*)(*U*)(*M*)(*PE*)(*D*)(.)(*I*)(*T*)(*J*)(*UM*)(*PE*)(*D*)(,)(*A*)(*ND*)
(*J*)(*UMPE*)(*D*)(,)(*A*)(*ND*)(*J*)(*UMPED*)(.)

(*O*)(*U*)(*T*)(*C*)(*A*)(*M*)(*E*)(*THE*)(*B*)(*AB*)(*Y*)(*B*)(*I*)(*R*)(*D*)

(')(*W*)(*H*)(*E*)(*RE*)(*I*)(*S* *M*)(*Y*)(*MO*)(*THE*)(*R*)(,')(*HE*)(*SAID*)(.)

(*HE*)(*L*)(*OO*)(*K*)(*E*)(*D*)( )(*FO*)(*R*)(*HE*)(*R*.)

(*HE*)(*L*)(*OO*)(*K*)(*ED*)(*U*)(*P*)(. *HE*)(*D*)(*ID*)(*N*)(*O*)(*T*)(*SE*)(*E*)(*HE*)(*R*.)

(*HE*)(*LOO*)(*K*)(*ED*)(*DO*)(*W*)(*N*)(. *HE D*)(*ID*)(*N*)(*O*)(*T*)(*SE*)(*E HER*.)

(')(*I W*)(*ILL*)( )(*G*)(*O A*)(*ND*)(*LOOK*)(*FO*)(*R*)(*HE*)(*R*)(' *HE*)(*SAID*)(.)

(*DO*)(*W*)(*N*)(,)(*O*)(*U*)(*T*)(*O*)(*F*)(*THE*)( )(*T*)(*RE*)(*E HE*)(*W*)(*EN*)(*T*.)

(*DOWN*)(,)(*DOWN*)(,)(*IT*)(*W*)(*A*)(*S A*)( )(*LO*)(*N*)(*G W*)(*A*)(*Y DOWN*)(.)

(*THE*)(*B*)(*ABY*)(*B*)(*I*)(*R*)(*D*)(*C*)(*O*)(*U*)(*L*)(*D NO*)(*T*)(*FL*)(*Y*.)

(*HE*)(*C*)(*O*)(*U*)(*L*)(*D NOT*)(*FL*)(*Y*)(,)(*BU*)(*T HE*)(*CO*)(*ULD*)(*WA*)(*L*)(*K*)
(.)(')(*N*)(*OW*)(*I*)(*W*)(*ILL*)( )(*G*)(*O A*)(*ND*)(*FI*)(*ND*)(*M*)(*Y*)(*MO*)
(*THE*)(*R*)(,)(' *HE*)(*SAID*)(.)

(*HE*)(*D*)(*ID NOT*)(*K*)(*N*)(*OW*)(*W*)(*H*)(*AT*)(*H*)(*I*)(*S M*)(*O*)(*THE*)(*R*)
(*LOOK*)(*ED LIK*)(*E*)(. *HE*)(*W*)(*EN*)(*T R*)(*I*)(*GHT*)(*B*)(*Y*)(*HE*)(*R*.)
(*HE*)(*D*)(*ID NOT*)(*SE*) (*E HER*.)

(*HE*)(*C*)(*A*)(*M*)(*E*)(*T*)(*O A*)( )(*KI*)(*T*)(*TEN*)(.)('*A*)(*RE*)(*YO*)(*U*)( )
(*MY*)(*MOTHE*)(*R*)(' *HE*)(*SAID T*)(*O*)(*THE*)( )(*KI*)(*T*)(*TEN*)(.)

(*THE KITTEN*)( )(*JU*)(*S*)(*T*)(*LOOK*)(*ED*)(.)(*IT DID NOT*)(*SA*)(*Y*)(*A*)
(*THI*)(*N*)(*G*.)

(*THE KITTEN*)(*WA*)(*S N*)(*OT*)(*H*)(*I*)(*S MOTHER*)(,)(*S*)(*O*)(*HE W*)(*EN*)
(*T*)(*O*)(*N*)(.)

(*THEN*)(*HE*)(*C*)(*AM*)(*E*)(*T*)(*O A*)(*HE*)(*N*.)('*A*)(*RE*)(*YO*)(*U*)( )(*MY*)
(*MOTHE*)(*R*)(' *HE*)(*SAID T*)(*O*)(*THE*)(*HE*)(*N*.)('*NO*)(,)(' *SAID T*)
(*HE HE*)(*N*.)

(*THE KITTEN*)(*WA*)(*S N*)(*OT*)(*H*)(*I*)(*S MOTHER*)(. *THE*)(*HEN WAS NOT*
HIS MOTHER.)(*S*)(*O*)(*THE*)(*B*)(*ABY B*)(*IR*)(*D*)(*W*)(*ENT*)(*O*)(*N*.)

(')(*I*)(*HA*)(*V*)(*E*)(*TO*)(*FI*)(*ND*)(*MY*)(*MOTHE*)(*R*)(' *HE*)(*SAID*)(. ')
(*BU*)(*T*)(*WHE*)(*RE*,)(*WHE*)(*RE I*)(*S S*)(*HE*)(, *WHERE*)(*CO*)(*ULD*)(*SHE*)
(*B*)(*E*)(')(.)

(*THEN*)(*HE*)(*C*)(*AME*)(*TO*)(*A*)(*DO*)(*G*.)('*A*)(*RE*)(*YO*)(*U MY MOTHER*' *HE*)
(*SAID T*)(*O THE*)(*DO*)(*G*.)

(')(*I AM*)( )(*N*)(*OT YOU*)(*R*)(*MOTHE*)(*R*.)( )(*I AM*)(*A DO*)(*G*)(,)('*SAID*
T)(*HE*)(*DOG*.)

(*THE KITTEN*)(*WA*)(*S NOT H*)(*I*)(*S MOTHER*)(. *THE*)(*HEN WAS NOT HIS*
MOTHER.)(*THE DO*)(*G W*)(*A*)(*S NOT HIS MOTHER*.)

(*S*)(*O THE B*)(*ABY B*)(*IR*)(*D W*)(*ENT*)(*O*)(*N*. *N*)(*OW*)(*HE C*)(*AME TO*)
(*A*)(*CO*)(*W*)(.)

('*ARE*)(*YO*)(*U MY MOTHER*' *HE*)(*SAID TO THE*)(*COW*.)

(')(*H*)(*OW*)(*CO*)(*ULD*)(*I*)(*B*)(*E YO*)(*U*)(*R MOTHE*)(*R*)(' *SAID THE*)(*COW*.)
('*I AM*)(*A*)(*COW*)(')(.)

(*THE KITTEN*)(*A*)(*ND THE B*)(*IR*)(*D W*)(*E*)(*RE*)(*N*)(*OT H*)(*I*)(*S MOTHER*)
(. *THE*)(*DO*)(*G*)(*A*)(*ND THE COW*)(*W*)(*ERE N*)(*OT HIS MOTHER*.)

(*D*)(*ID*)(*HE*)(*HA*)(*VE A*)(*MOTHE*)(*R*.)

(')(*I*)(*D*)(*ID*)(*HA*)(*VE A MOTHER*)(' *SAID THE*)(*B*)(*ABY BIRD*)(. ')
(*I*)(*K*)(*N*)(*OW*)(*I*)(*D*)(*ID*)(. *I HA*)(*VE*)(*TO*)(*FI*)(*ND*)(*HE*)(*R*.)
( )(*I W*)(*ILL*)(. *I*)(*W*)(*ILL*)(.')

(*NOW*)( )(*THE B*)(*ABY BIRD D*)(*ID NOT*)(*WA*)(*L*)(*K*)(. *HE*)(*RA*)(*N*.)
(*THEN*)(*HE*)(*SAW*)(*A*)(*C*)(*AR*.)( )(*CO*)(*ULD*)(*TH*)(*AT*)(*O*)(*L*)(*D*)
(*THIN*)(*G*)(*B*)(*E*)(*H*)(*I*)(*S MOTHER*.)(*NO*)(,)(*IT CO*)(*ULD*)(*NO*)(*T*.)

(*THE B*)(*ABY BIRD D*)(*ID NOT S*)(*T*)(*O*)(*P*)(. *HE*)(*RAN*)(*O*)(*N*)(*AND O*)(*N*.)

(*NOW HE*)(*LOOK*)(*ED*)(*WA*)(*Y*,)(*WA*)(*Y*,)(*DOWN*)(. *HE*)(*SAW A B*)(*O*)
(*AT*)(. ')(*THE*)(*RE*)(*SHE I*)(*S*' *SAID THE*)(*BABY BIRD*)(.)

Fig. 7

particular string was last parsed. The probable mean rate of change of the count associated with a particular string over a long period during which no new strings are added to the library, however, is $p_i m - 1$ where p_i is the probability that this particular string i, rather than any other string in the library, will be parsed, and m is the number of strings in the library. Herein lies the justification for making the amount by which the string counts are updated and the threshold above which compound sequences are formed vary with m, the number of strings in the library. By using the updating method chosen the CRL will only stagnate if for all the strings p_i is equal to, or less than, $1/m$, but since there are m strings in all:

$$\sum_{i=1}^{m} p_i = 1$$

and

$$p_i \leqslant \frac{1}{m}$$

therefore

$$p_i = \frac{1}{m}$$

—thus stagnation can only occur if all the strings are equiprobable. Having chosen to update the string counts by m when they occur, the threshold above which compound sequences are formed must be greater than m since otherwise every string parsed would immediately be formed into a compound string. The value of $2m$ was chosen as a threshold intuitively, and found to be very effective.

When compound strings are formed they could be formed in one of four ways (assuming that it is also required to ensure that any string formed has actually occurred at least once, at the moment of its formation); by adding the preceding symbol, by adding the following symbol, by adding the preceding string, or by adding the following string. It was decided that it was always more efficient to add a string rather than a symbol because the compound strings would then get longer more quickly. It was also decided that, in the absence of any *a priori* knowledge of the language structure, it was equally efficient to use either the preceding or the following string and that it was easier to use the preceding string since the information was already available.

No mention has yet been made of what happens to the strings in the library whose counts become very low (very highly negative). Consider a situation on which a string *ABCD* gets joined onto a string *EFG* when the CRL is acting on a language in which the only occurrence of the string *ABCD* is in a string *ABCDEFG*. In this case the count for the string *ABCD*, after the string *ABCDEFG* has been formed, will always thereafter get more and more negative. It is evident that there is no longer any point in retaining the string *ABCD* in the library. However, there is no way of telling, in the absence of external information about the language, that this really is the case. The problem is, therefore, how negative should the string count be allowed to become before the string is rejected from the library? The procedure currently used is that when the counts become less than $-2m$ the strings are rejected. The only justification for this pro-

(HE)(CA)(LL)(ED)(TO THE BO)(AT)(, BU)(T)(THE B)(O)(AT)(D)(ID NOT S)
(TO)(P)(. THE B)(O)(AT)(W)(ENT O)(N.)

(HE)(LOOK)(ED WAY,)(WA)(Y)(U)(P. HE)(SAW A B)(I)(G)(P)(L)(AN)
(E. ')(HE)(RE I)(A)(M,)(MOTHER)(,)(' HE)(CA)(LLED)(OU)(T.)

(BU)(T)(THE)(PL)(AN)(E DID NOT S)(TOP. THE)(PLANE W)(ENT ON.)

(JUST)(THEN)(, THE B)(ABY BIRD SAW A BI)(G)(THIN)(G.)( )(THI)(S M)
(U)(S)(T)(B)(E)(H)(I)(S MOTHER. ')(THE)(RE)(SHE IS)(' HE)(SAID)
(. 'THE)(RE I)(S M)(Y)(MOTHER)(')(.)

(HE)(RAN)( ) (RI)(GHT U)(P TO)(I)(T.)('MOTHER,)(MOTHER)(. HERE I)
(A)(M, MOTHER' HE)(SAID TO THE BI)(G THIN)(G.)

(BUT)(THE B)(I)(G THING)(JUST)(SAID)(, ')(S)(NORT)('. ')(OH,)
(YO)(U ARE NO)(T)(MY)(MOTHER)(,)(' SAID THE)(BABY BIRD)(. ')
(YO)(U ARE)(A S)(NORT)(. I HA)(VE)(TO)(G)(ET)(OU)(T O)(F)(HE)(RE'.)

(BUT THE B)(ABY BIRD)(CO)(ULD)(NOT)(G)(ET)(AWA)(Y. THE)(SNORT)
(W)(ENT)(UP)(.)(IT)(WENT)(WA)(Y, WA)(Y)(UP.)(AN)(D UP)
(,)(UP, UP)(W)(ENT)(THE B)(ABY BIRD)(.)

(BUT)(NOW)(, WHERE)(WA)(S)(THE)(SNORT)( )(G)(OI)(N)(G.)(')
(OH, OH.)(OH)(. W)(HA)(T)(I)(S)(THI)(S S)(NORT)(G)(OIN)(G TO)
(DO)( )(TO)(ME.)(GET)(ME)(OU)(T O)(F HE)(RE'.)

(JUST)(THEN THE SNORT)( )(CAME TO)(A S)(TOP.)

(')(WHE)(RE)(AM)(I)(' SAID THE)(BABY BIRD. ')(I WANT)(TO G)(O)
(HO)(ME.)(I WANT)(MY MOTHER)(.')

(THEN S)(O)(ME)(THING)(HA)(P)(PE)(N)(ED)(. THE)(SNORT)(P)(U)(T)
(TH)(AT BABY BIRD)(RI)(GHT)(BA)(C)(K)(IN)(THE)(T)(RE)(E)
(. THE BABY BIRD WA)(S)(HOME)(.)

(JUST)(THEN THE)(MOTHER)(B)(IR)(D)(CAME)(BAC)(K TO THE T)(RE)
(E. ')(DO)( )(YO)(U K)(NOW)(WHO)(I)(AM)(')(SHE)(SAID T)(O)(HE)
(R BAB)(Y.)

(')(YE)(S)(,)(I)(KNOW)(WHO YO)(U A)(RE)(,' SAID THE)(BABY BIRD. ')
(YO)(U ARE NOT)(A)(KI)(T)(TEN)(. YO)(U ARE NOT)(A)(HEN)(. YOU
ARE NOT A)(DOG.)( )(YO)(U ARE NOT)(A)(COW. YOU ARE NOT A)(BO)
(AT.)(O)(R)(A)(PLAN)(E)(. O)(R A SNORT. YO)(U ARE A B)(IR)(D,)
(AND)(YOU ARE)(MY MOTHER)(.')

Fig. 7 continued

cedure is expediency, but it should be noted that any string so rejected must have a low frequency, and also that it can always be built up again if necessary.

Two major ways of increasing the power of the CRL procedure in particular applications have also been considered, these involve the association of further properties with the strings, and the extension of the CRL to deal with more complex language structures.

In particular applications it might be desirable to associate further properties with strings of symbols apart from the actual symbols themselves. For example, suppose that the symbols consist of operation codes and the text consists of a computer program. It is, for example, a property of the strings, rather than of the symbols, that the first two operation codes in the string share the same first operand. It might be desirable that the library contains two strings with identical sequences of symbols but for one of which this property holds and for the other of which it does not. Thus by using the same basic CRL procedure further particular information about the text under study may be extracted.

The CRL procedure may also be extended to deal with more complex language structures. It is evident that the basic CRL procedure described above would be unable to detect certain more complex language structures. For example, in the English language, it would never be able to detect that an 'open inverted commas' is always paired with a 'close inverted commas'. Suppose however that an extended CRL procedure was designed that consisted not only of a set of strings but also of a set of sets, and that set numbers may appear in the strings, and that the sets consist of sets of string numbers. An example is illustrated in **Fig. 8.**

SETS	STRINGS
1: 1,4 2: 2,4 3: 3,1	1: *AB* 2: 1*C* 3: *D*3*E* 4: *A*

Fig. 8. Extended CRL

A
AB
AC
ABC
DABE
DDABEE
DDDABEEE
. . . etc.

Fig. 9. Expansion of Fig. 8

This library would be translated as a set of strings whose elements are either symbols as stated, or else any of the strings whose numbers appear in the set whose number is stated. In the example shown the set of possible sequences that may be generated from the library is shown in **Fig. 9.**

It may be shown that such a system would be able to parse any syntax that it is possible to state in Backus Naur form (Backus, 1959). An extended version of the CRL may be designed, by suitable alteration of the parsing, updating, and new string formation procedures, which is able to build up such a library in the same manner as the simple CRL library, though at a proportionately slower rate.

5. Applications of the CRL

The CRL procedure has two main areas of application: its use to discover recurrent patterns in linear strings of symbols and its use, given such a library, to parse such strings of symbols. Much time has been spent, for example, trying to 'fingerprint' the writings of Shakespeare in an attempt to prove that they were written by Bacon. This has been done, basically, by the use of word frequency counts. The CRL would probably provide a better procedure for doing this, in that it involves no *a priori* assumptions, and the resulting library contents would provide a much more natural 'fingerprint' of the writing.

As a less academic example, consider newspaper offices transmitting their stories across transatlantic cables (assuming that the possible errors due to noise have been dealt with by suitable coding). Since such newspaper stories are in a highly redundant code (the English language) it would be advantageous to be able to code the data more efficiently so as to reduce the transmission time. Suppose, then, that the stories are first fed into a CRL and transformed, by the parsing system, into a code in which each symbol string s_i is replaced by its number i. If the CRL is kept constantly ordered so that, for any i:

$$c_i \geqslant c_{i+1}$$

—where c_i is the count for string s_i, and if the numbers i are coded so that it takes less time, in general, to transmit the number i than number $i + 1$, then the CRL coding will automatically tend to an optimal information/unit time coding. Thus by coding with a CRL, transmitting the string numbers, and de-coding with an identical CRL, the transmission time would be minimised automatically.

6. Acknowledgements

The author's thanks are due to W. Collins and Sons Ltd. and to Mr. P. D. Eastman for permission to quote from the text of their delightful children's story, 'Are You My Mother?'.

References

Backus, J. (1959). The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM–GAMM Conf. Inf. Processing. *Proc. ICIP*, UNESCO, Paris, pp. 125–132.

Eastman, P. D. (1962). *Are You My Mother?* London: W. Collins and Sons Ltd.

Michie, D. (1967). Strategy Building with the Graph Traverser, *Machine Intelligence 1*, pp. 135–152. Collins, N. L., and Michie, D. (Eds.). Edinburgh: Oliver and Boyd.