STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

REKENAFDELING

Report R 628

Sorting by Chain Formation

A Program for a Computer Handling Punched Tape

by

J.A.Th.M. van Berckel

November 1962

The Mathematical Centre at Amsterdam, founded the 11th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for Pure Research (Z.W.O.) and the Central National Council for Applied Scientific Research in the Netherlands (T.N.O.),by the Municipality of Amsterdam and by several industries.

Introduction

This report gives a description of a program made at the request of Dr. F. de Tollenaere, editor of the "Woordenboek der Nederlandsche Taal", Leyden, The Netherlands. The program was written for the X1 computer of the Computation Department of the Mathematical Centre, Amsterdam. The output consists of:

1. A Word Index: an alphabetical list of words with references to their locations in the text.
2. A Frequency List: a list of words arranged in order of descending frequency; words with the same frequency stand in alphabetical order.
3. A Length List: a list of words arranged in order of descending word length; words with the same number of characters stand in alphabetical order.
4. A Reverse Index: a sorted list of words, all words alphabetized from the ends of the words backwards.

As a first experiment we used the Dutch sentences from the book "Tyrocinium Linguae Latinae" (1552) by Apherdianus. A reproduction of this book together with the results delivered by the X1 will be published soon by Mouton & Co., Publishers, The Hague.

Since the input to the X1 of the Mathematical Centre is effected by means of a high speed photo-electrical punched tape reader [1] and the output by a 7-holes tape punch and an electrical typewriter, it was an interesting problem to find, for a computer fitted with this equipment, a compromise between memory space and computing time. Other input and output apparatus had already proved their applicability for this problem elsewhere. In spite of their disadvantages they are in some respects to be preferred to punched tape, e.g. magnetic tape which, owing to the fast accessibility of a great amount of information, is more useful as a temporary extension of the memory than punched tape [2]. But we had to

---

1) Maximum speed of tape reader, tape punch and typewriter: 1000, 25 and 10 symbols/second resp.

2) Here, we have in mind the preparation of a concordance where for each occurrence of a word, the context must be memorized instead of its reference.

limit ourselves to the equipment of a fast scientific computer [1] as mentioned above, although we hope to have a high speed 7-holes tape punch at our disposal after some time (300 symbols/second).

Input and Output Tapes

In order to punch the input tape and to print the results (delivered by the X1 on output tape) use is made of a Flexowriter. This is an electrical typewriter handling punched tape. The manipulation of a key has two effects:
First, the typewriter reacts as any common typewriter would. Second, a pattern of holes corresponding to the typed symbol is punched in a paper tape. Also each manipulation that performs a typewriter setting corresponds to a combination of perforations (heptade) in the 7-channel tape [2]

In the Flexowriter punching code only those heptades that have an odd number of holes are used. This fact would halve the $2^7 = 128$ possibilities, if there was no application of "case definition", i.e. Upper Case or lower case, restoring this number of possibilities nearly completely. The 52 upper and lower case letters, the 10 digits, the punctuation marks, the apostrophe, the brackets etc. can easily be represented in this way. Some odd heptades have not even been used. In typing a text this is printed on the paper and besides that a true copy of it is punched in the tape. This punched copy can be reproduced and amended by means of the same apparatus. The tapes produced in this way can be read by the tape reader of the X1.

---

1) Scientific, as contrary to administrative computer. We give here some specifications of the X1 of the Mathematical Centre:
Memory space: 12288 addresses core memory of 27 bits each.
Computing speed:

Addition, subtraction, fast multiplication by 10 and logical multiplication:		64	micro sec.
Multiplication and division:		500	" "
Absolute jump order		36	" "
Subroutine jump		72	" "
Shift over n positions	40 +	8n	" "

2) These are: SPACE, TAB, New Line Carriage Return, Upper Case, lower case and STOPCODE.

The Consideration of Various Methods with regard to Running Time

Considering the maximum speed of the reader and the tape punch, it will be clear that the total computing time largely depends on the speed of the tape punch. If we did not take special measures, then during the execution of a punching program the X1 would continually be wasting much time in waiting for the next heptade to be punched. This is the case when the following procedure is used:

All words from a text beginning with the letters a through f are alphabetized, after which the list obtained in this way is offered to a punching program. This will deliver the first part of the results list. After the punching has been finished a second run is made alphabetizing the words with the initial letters g through k, etc.

Cutting the text in two or more "slices" (the first containing all words with the initials a through f, the next g through k, etc.) is necessary because the memory space is often insufficient to hold all different words of the text at the same time [1].

The procedure outlined above is very inefficient, in view of the possibilities of the X1. The solution of this problem has to be found in the interruption facility of the X1. In brief the thing boils down to this [2]: A "read", "punch" or "type" order being given, the apparatus in question does not accept another order during a fixed time - 1, 40 and 100 milliseconds respectively. The program can utilize this time by continuing to sort, until the so-called interruption signal of the apparatus concerned indicates that a new communication order can now be executed. At the moment of the interruption all data necessary for the continuation of the sorting program are stored, after which the interruption program gives the new communication order and jumps back to the main program as soon as possible, restoring the status quo ante.

In order to profit most by this facility the program should perform the following items:

---

1) Compare the method followed in: A Word-Index to Racine's Mithridate, [2.].

2) Treated at length in: Communications with an Automatic Computer, [1].

1. Reading and alphabetizing the first slice (e.g. a through f).
2. At the end of a run offering the slice to an interruption program which completely controls the punching.
3. Reading and alphabetizing the next slice during the waiting times of the tape punch. (The tape reader and the sorting program are so fast that this will be completed before the interruption program has finished its job.)
4. Repeating items 2 and 3, until the last slice has been read, alphabetized and punched.

The "size" of the slice must be defined before each run; it is sufficient to put the "upper bound" into the console.

In order to enable the interruption program to provide for the output, special arrangements have to be made for storing the information. We might reserve a section of the memory, on which the punching program can draw at the moments suited to the tape punch; but the use of such a fixed part of the memory has many disadvantages:

1. All information has to be transported again. This will take much time and will increase the chance of making errors.
2. Another part is taken from the memory space, which is too small already for this kind of work.
3. Such a fixed reservation is either too large - another waste of memory space - or too small. In the last case the main program will have to wait for the punching program to release enough space.

To get round all these disadvantages we may declare "punch store" that section of the memory in which a slice has been built up during the last run. The punch store made in the previous run not yet being emptied, the new one has to be coupled to it. All this can easily been realized by "chain formation". We shall return to this subject below.

The "References" Program

Since there is no instruction "alphabetise" in the order code of the X1, all this has to be simulated by arithmetical operations. Just as the Flexowriter interprets a combination of perforations as an order to perform a mechanical operation, so the X1 takes this heptade as a number.

The characters in the Flexowriter code do not have such a numerical value that the comparison of words can be done by simple arithmetical operations. That is why all tape symbols are translated into a more manageable code as soon as they enter the computer. All letters get a value less than 32 [1], and so we can represent each one in 5 bits and put 5 letters into a 27-bits address. In this way the word as a whole is represented by a number depending entirely on the letters from which it is composed. Henceforth we shall often use the concept "word" when we mean the number that represents the word.

During the translating procedure the program checks if the heptade just read has an odd number of holes and whether it really exists in the Flexowriter code [2]. By this check errors in tape reading will be detected too.

Alphabetizing can be done by subtracting the word just entered (n') from the words already stored in the memory [3]; if the result of this subtraction of n' from word n is zero, we have to append the reference of n' to n. If n' is less than n we have to put n' before n else, if greater, we have to repeat both tests, defining n as the next word from the memory.

In order to insert a word into the list we could have shifted all words beginning with n over as many positions as needed for storing n' with its reference. In this way we do not waste memory space, but it takes very much time and this time increases as the list grows longer. Besides this these shifts are necessary in the case of equality of words too because the reference must be inserted into the list.

---

1) a = 1, b = 2,....., z = 26, ' (apostrophe) = 27.

2) Undefined heptades can be caused by the coincidence of two heptades the Flexowriter not having transported the tape. If such a coincidence generates a defined Flexowriter symbol, we have to rely on the "proof-reading" to detect this mistake.

3) J.A. Painter discusses an alternative procedure in his "Computer Preparation of a Poetry Concordance", [3] : Reading and alphabetizing are not executed simultaneously. "The first phase scanned the line of poetry from the input cards. It broke the line into its component words and appended an identification of the source line. After all the lines of text had been scanned the words were sorted into alphabetical order. This was phase two". He had at his disposal 5 magnetic tape units, but said about phase 2 "This phase was the most time consuming". Where no magnetic tape units are available, as in our case, this procedure means an endless job as we experienced in the work described in Report R 642 "Onderzoek Woordfrequentie" (= Word Frequency Research), Math. Centre, Nov.1961, Amsterdam.

In this program the system of "chain formation" has been introduced. For this purpose to every word in the store the address of the following word in alphabetical order has been appended. This is called: "Pointer Alphabetical Successor" - PAS [1]. Inserting a word n' between n_{k-1} and n_k looks as follows: Store n' in the first "empty word unit" and append the address of n_k to n' and the address of n' to n_{k-1} (the address of n_k was appended to n_{k-1}). In this way we can build up a "word chain" in which the words are scattered through the memory, but we can reach each word of the chain by using "B-corrected" orders [2]. This technique saves a lot of time, the shifting not being necessary anymore. It is more expensive in memory space as every word has to carry this extra information.

At the start of the program a part of the memory space has been divided into "empty word units" of 4 addresses. To get an "empty word chain" these units are linked among themselves by pointers. In the "Pointer Empty Word Chain" - PEWC - the first address of this chain is kept and in the "Counter Empty Word Chain" - CEWC - the number of units in the chain is recorded. In order to store a new word, the first empty word unit must be cut from the empty word chain by transporting to PEWC the address of the second unit (which causes it to be the first one from now on) and by subtracting 1 from the CEWC (The address of the second empty word unit was appended to the first). If a word is longer than 9 letters, the 10th and following letters up to the 29th are stored in a "successive unit", for which purpose the next empty word unit is taken from the empty word chain; modifying PEWC and CEWC accordingly. To the word unit containing the first 9 letters we have to append the address of this successive unit. If a word is longer than 29 letters it is chopped off behind the 29th letter.

---

1) For a list of abbreviations used see page 15.

2) The B - register can be used as an "address modification register": In the case of a B - correction to the address part of the order the contents of the B - register are added immediately after this order has been read from the store, it is then executed with the modified address. The B - register consists of 16 bits and we have to reserve, therefore, 16 bits for the various pointers we use, although a reservation of 14 bits would have been enough to reach the approximately 10000 addresses of the working space.

In order to reduce the searching time still further, the program constructs several "initial chains", viz. so many chains as initial letters are allowed in t h i s run. For each initial chain is a "Pointer Initial Chain", depending on the initial letter i" - PIC_i. When a word with initial letter i has been read, the PIC_i refers it to the initial chain it belongs to. After each run these initial chains are coupled behind each other by appending the first address of a chain to the last word of the previous chain. The total chain obtained in this way can act as "punch store" and is now called "output chain". Eventually, this output chain must be coupled behind the one made in the previous run.

The treatment of the references to the text needs some explanation too. A fixed reservation as in "A Word-Index to Racine's Mithridate" [2] encountered several difficulties. There, corresponding to each word room was made for up to 64 references, implying waste of memory space for many words, while it is often insufficient for words with a high frequency. Since dr. de Tollenaere needed a l l references of these frequent words, here, too, application of chain formation proved to be effective.

The first time a word occurs, to the word unit containing the first 9 letters the reference is appended, using 11 bits of the second address in the word unit. The remaining 16 bits are reserved for the "Pointer Reference Successor" - PRS. In this way we can construct a "reference chain", if it appears that a word occurs more than once. Use is made of an "empty reference chain", on which can be drawn in a similar way as outlined for the words, but with application of "Pointer Empty Reference Chain" and "Counter Empty Reference Chain" - PERC and CEWC respectively. The "empty reference chain" consists of units of 1 address and is made at the start of the program too.

A sort of "short-circuit" has been built in to avoid the program running through an ever-increasing list of references again and again when handling frequent words. This short-circuit only applies to words shorter than 6 letters, as most frequent words are. If a word has 6 letters and the last bit of this 6th letter is equal to zero, short-circuiting is possible too, viz. the 6th letter must have an even

numerical value - b, d, f, etc. In the space formed by this bit together with the 15 bits reserved for the 7th, 8th and 9th letter we can append a pointer to the last unit of the reference chain . After each occurrence of the word the address of the new reference must be put in this space. Of course we have to append this address to the last unit of the reference chain too. Figure 1 shows schematically this complex of chains and figure 2 gives the arrangement of the word unit containing the first 9 letters.

The Output Program

As already mentioned by the way, an interruption program once activated can control the output, independent of a main program. The latter only has to supply the material. In our case, the output chain consists of the coupled initial chains with "branches" consisting of reference chains appended to most of the words. The "Pointer Output Chain" - POC - refers to the first address of this output chain.

As soon as the punching program has got so far that the next word can be punched, the word unit is found with the aid of POC and the information (i.e. the contents of this unit) is transported to a small "output store". In this auxiliary store the information of a successive unit, if any, is put after that of the first one. A reservation of 8 addresses for this output store will be enough.
We now have to release from the output chain the word unit(s), the contents of which have just been transported; this is effected by transporting to POC the PAS of the word, which is contained in the first unit, and we have to connect the word unit(s) to the empty word chain, with suitable modifications of PEWC and CEWC.
The interruption program translates the letters of the word into Flexowriter code and consecutively punches them. During each waiting time of the tape punch the X1 gets through a part of the main program.

After all letters have been punched the program has to run through the reference chain to punch the references. This chain starts in the second address of the word unit containing the first 9 letters. Following reference units, if any, give rise to modifications of PERC and CERC. In

this way the output chain is emptied and so we get a punched tape from which the Flexowriter can print a list.

Since the punching program interrupts the main program at moments unknown beforehand and then perhaps modifies PEWC, PERC, POC, CEWC or CERC, it is necessary to prevent this interruption in parts of the main program in which those data are consulted or brought into agreement with one another. All this can be achieved by an order prohibiting this special interruption during a specified part of the main program. The same order has been put at the beginning of the punching program preventing that program from interrupting itself.

During the manipulation of a text the punching of a previous slice may be lagging behind the sorting process so much, that at some moment the empty word chain or empty reference chain has been exhausted. That is why the main program asks at the opportune moments: "Enough space available?" If the answer is in the negative, the main program waits in a cycle, while the punching is continued in the normal way. In this cycle the question is repeated every time, until an affirmative answer ends it, because the punching program has released an information unit. In this case the main program is continued by storing the new information.

It may happen, however, that the punching program comes to a stop, because the previous slice has been punched completely. If at that moment the question: "Enough space available?" has not been answered with "Yes", then obviously the slice now being sorted is too large, which means that we allowed too many initial letters (i_1 through i_k) to be sorted in this run. This is an operating error; it implies a considerable waste of time, because the main program has been sorting words that have to be "deleted" now and, this being done, the remaining part of the text still has to be read. The program automatically performsthe deletion mentioned, thereby constructing a new empty word chain and empty reference chain. Running on, it will now confine itself to words with initial letters i_1 through i_{k-1}. As soon as the 27th initial letter (= apostrophe) has been worked up, the main program is waiting in a cycle until the output chain has been punched; then the X1 stops.

Besides the tests incorporated into the program to detect errors in tape reading or to prevent operating errors, there are several that ensure correct operation of the program. The last mentioned have proved their usefulness during the testing of the program. If the conditions were not satisfied then the X1 stopped. At the end of an input tape the X1 stops reading only, the punching continuiing at all times; when the next tape has been put into the reader the reading process can be continued by means of the reversing of a switch on the console.

For the output of information the typewriter is utilized too, for which purpose of course use has been made of standard printing routines. Since these are interruption programs too, it is necessary to take the numbers to be printed from the output chain in order to guarantee a good working. The program achieves this by separating from the empty word chain, in the way described above, a word unit, in which the number and also a printing code [1] are put, after which this "print unit" is connected to the output chain. An indication "special output" is appended to this unit too, by which the interruption program can detect that the information in this unit has to be printed instead of punched.

The printing of data mainly serves for controlling purposes, as for instance the printing of all page numbers. By this the operator can check the input tapes being read in good order and, therefore, it is desirable that this information is printed as soon as it has been read from the tape. This is achieved by the insertion of the print unit into the output chain before "the punch units" but behind "print units" that may still exist. At the end of each page a number is printed giving the total amount of words which has been read till then.

For the sake of the punching of the input tapes certain conventions have been adopted. A code number is printed as soon as the X1 detects that the tapes are not in accordance with these conventions. This number tells something about the kind of error and indicates the line number in

---

1) The printing code indicates the way of printing of the number - absolute value or with sign, with or without punctuation, etc. Instead of a printing code the starting address of a subroutine can also be given in this space; by this it is possible to print text for instance.

the text where it has been made.

The "Frequencies" and "Lengths" Programs

The output tapes with the results of the "references program" act as input for the programs that have to make the lists in which the words are sorted in order of descending frequency or length respectively.[1]. This procedure has the great advantage that no alphabetizing is needed anymore if it appears that words have the same frequency. Since in this "frequencies program" no references have to be manipulated the space reserved for them in the references program (second address of a word unit) can be used to store the frequency together with a pointer [2].
In this program too, the material must be cut into slices. For instance: In the first run: maximum frequency through 10; second run: 10 through 2 and in the third run words with frequency 1. For this purpose, we have to put each "lower bound" into the console.

The determination of the length of a word is not difficult; the frequency of a word is found by counting the number of references. With this number a search is made in the "frequency chain" [3]. Just as in the references program, room has been made in the word unit in which we can put PAS, the pointer to the alphabetical successor which, in this case, is a word having the same frequency. If PAS = 0, in other words: if there is no alphabetical successor with the same frequency, then in the second address of the word unit a pointer has been put giving the address of the first word unit of the next alphabetical chain. This is called "Pointer Frequency Successor" - PFS. If PAS $\neq$ 0, then this space indicates the address of the last unit in the chain of words with the same frequency: "Pointer End of Alphabetical Chain" - PEAC and then PFS is appended to the last word of this chain. The schemes in figures 2 and 3 will illustrate all this.

---

1) For both problems almost the same program can be used. Where we speak of frequency in this description one could just as well read length.

2) Of course the empty reference chain has been dropped too.

3) Just as in the references program we do not discuss the details related to the "start" of the chain.

The searching process acts as follows:
With the aid of the "Pointer Frequency Chain" - PFS - the first word with the highest frequency is found in the store. If the frequency of the word just entered - f(n') - is less than that of the word in the store, then we have to find the alphabetical chain of words with the next lower frequency. Depending on the value of PAS, the PFS is found either in the first and only word unit or in the last unit of the chain. After the PFS has been read into the B - register, f(n') can be compared with the frequency of the next alphabetical chain etc.

Considering the case in which the frequency of the word just read is equal to the frequency of the word with which it is compared, then owing to the fact that the words are read from the input tape in alphabetical order, we know that the word must be connected at the end of t h i s alphabetical chain. Now, there are two possibilities:

1. The alphabetical chain consists of only one word n (PAS = 0 and the PFS has been stored in the word unit containing the first 9 letters). After storing n' and f(n') in the first word unit(s) from the empty word chain - modifying PEWC and CEWC - we have to transport the PFS to n', while its PAS must be made equal to zero. Next we have to replace the PAS of n by the address of n'. The PFS of n is replaced by the address of n' too, but it is called PEAC from now on. See also figure 3, chains c and b.

2. The alphabetical chain consists of more units, PAS $\neq$ 0. With the aid of PEAC, appended to the first word unit n_1, we can find the last word n_k of the chain. Having transported the PFS from n_k to n', we must replace the PAS of n_k and the PEAC of n_1 by the address of n'. (Figure 3: b and d or d and a).

Let us now take the case that f(n') is less than the frequency in chain c_k but greater than the frequency in chain c_{k-1}; in other words a new chain has to be inserted between two existing ones. The PFS in chain c_k is found in the way as described above (PAS = 0: then in the first word; PAS $\neq$ 0: then in the last word with the aid of PEAC). The word to which this PFS is appended is called $n(f_k)$ from now on, and

the first word of chain c_{k-1} is called $n(f_{k-1})$. After storing n' and making its PAS equal to zero we must transport to n' the PFS appended to $n(f_k)$, this is the address of $n(f_{k-1})$. Having done this, we only have to replace the original PFS of $n(f_k)$ by the address of n', after which the frequency chain is in good order again.

In order to reduce the searching time it would have been better to construct the list from low to high, because there are more words having a low frequency than words having a high one and also more short words than long words. But this would meet with several difficulties with respect to the output, because the chain can be run through in just one direction and the order required us to deliver a list of descending frequencies.

The Reverse Index Program

This 4th program runs nearly parallel with the references program. Each word is reversed as soon as it has been read and in this way it is sent into the "alphabetizing routine". Of course the punching program has to provide for the turning back.

Notes

During the drawing up of the program the type area which the publisher wants to use in the publication has been taken into account, so that the production can be accomplished by means of a photographical reproduction process.

Before the text was punched, dr. de Tollenaere had the homographs [1] provided with the diacritical figures needed. This took a lot of time and trouble. In the future these separations will be made afterwards by using a tentative list with references by which one can get a better general view of all words. Then we may as well "lemmatize" too. This is the joining under one heading of all occurring forms of a word depending on difference in case, number, declension or conjugation; also, in spellings

---

1) Homographs are words with an equal notation, but with different meanings.

that are not normalized it is the joining of all different notations of a word.

Provisions are made to join the component parts of a verb which have been separated. The separation of verbs "separable composed" is a very common construction in Dutch. The program can also manage the possibility, not entirely theoretical, that between the two parts of a verb thus separated a construction of the same kind occurs again. For instance, the sentence: "Ik kwam hem, maar dat viel mij eerst niet op, elke dag tegen". In English: "I met him every day, but that did not strike me at first"; tegenkomen = to meet and opvallen = to strike.
It will be clear that the respective component parts must be "nested", otherwise the wrong parts would be coupled to each other.

We needed 1 hour and 12 minutes computer time to make the word index from a text of 11 150 words; herein, of course, the time needed for testing the program is not included.

Abbreviations

REF Reference: Location of a word in the text

FREQ Frequency: Number of occurrences of a word

LENG Length : Number of characters in a word

Pointers and Counters in the working space

PEWC Pointer Empty Word Chain: indicating the address of the first unoccupied word unit.

CEWC Counter Empty Word Chain: indicating the number of the unoccupied word units.

PERC Pointer Empty Reference Chain: indicating the address of the first unoccupied reference unit.

CERC Counter Empty Reference Chain: indicating the number of the unoccupied reference units.

PIC_i Pointer Initial Chain, depending on the initial letter (i): indicating the address of the first word unit of the chain of words with the initial letter (i).

PFC Pointer Frequency Chain: indicating the address of the first word unit of the frequency chain = the word with the highest frequency.

POC Pointer Output Chain: indicating the address of the next word unit whose contents must be punched or typed.

Pointers in the information units

PAS Pointer Alphabetical Successor: indicating the address of the next word in the alphabetical sequence.

PRS Pointer Reference Successor: indicating the address of the next reference unit belonging to the word.

PFS Pointer Frequency Successor: indicating the address of the first word unit of the alphabetical chain of words with the next lower frequency.

PEAC Pointer End of Alphabetical Chain: indicating the address of the last word unit in the alphabetical chain of words with the same frequency.

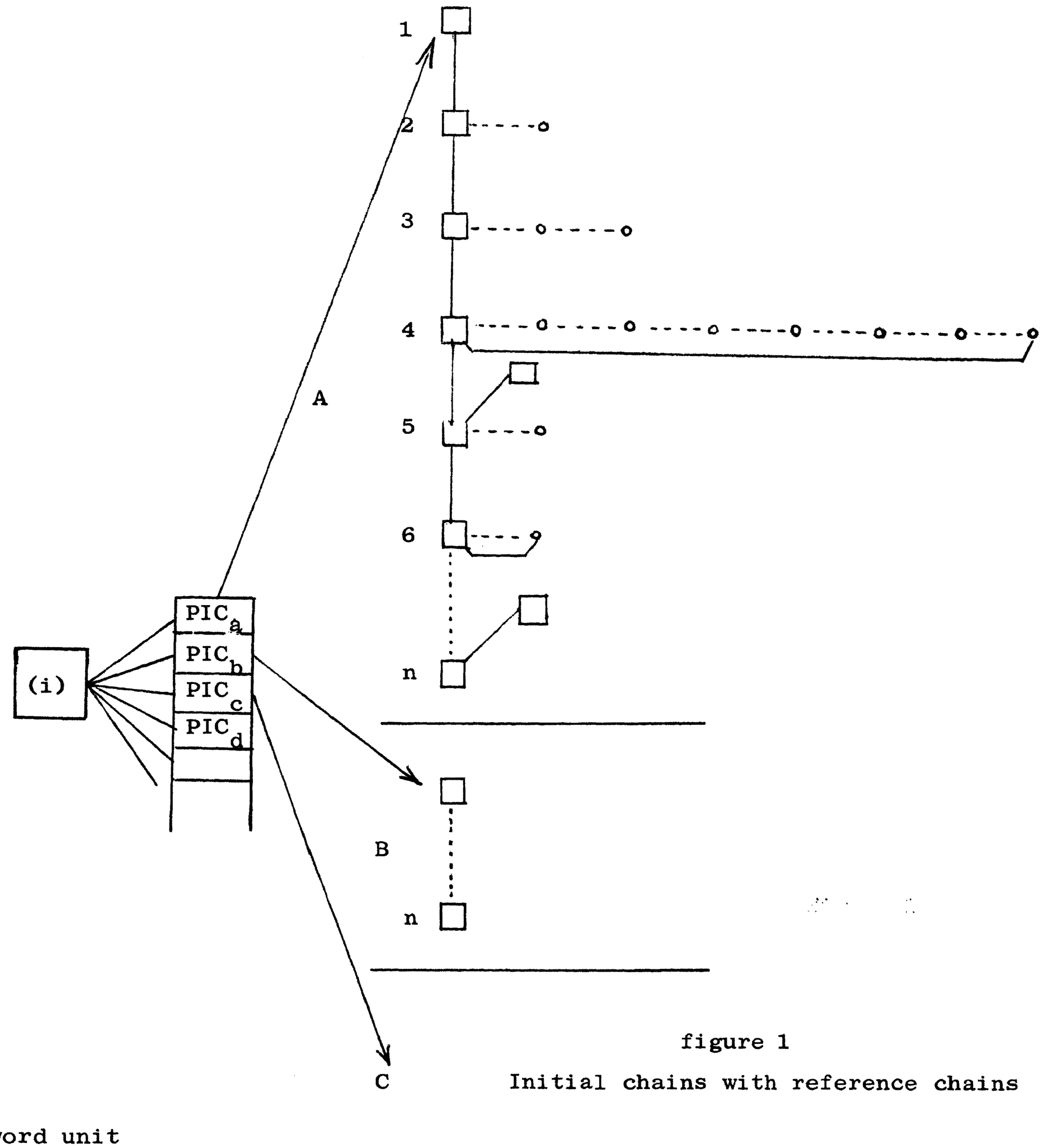

figure 1

Initial chains with reference chains

□ = word unit

o = reference unit

Depending on its initial letter (i), a word is referred, with the aid of PIC_i, to the first address of the initial chain A, B, C, etc.

A1 and An have just one reference stored in the word unit containing the first 9 letters.

A5 and An have more than 9 letters which are stored in a successive unit.

Short-circuiting could be employed in the words A4 and A6, these words being shorter than 6 letters.

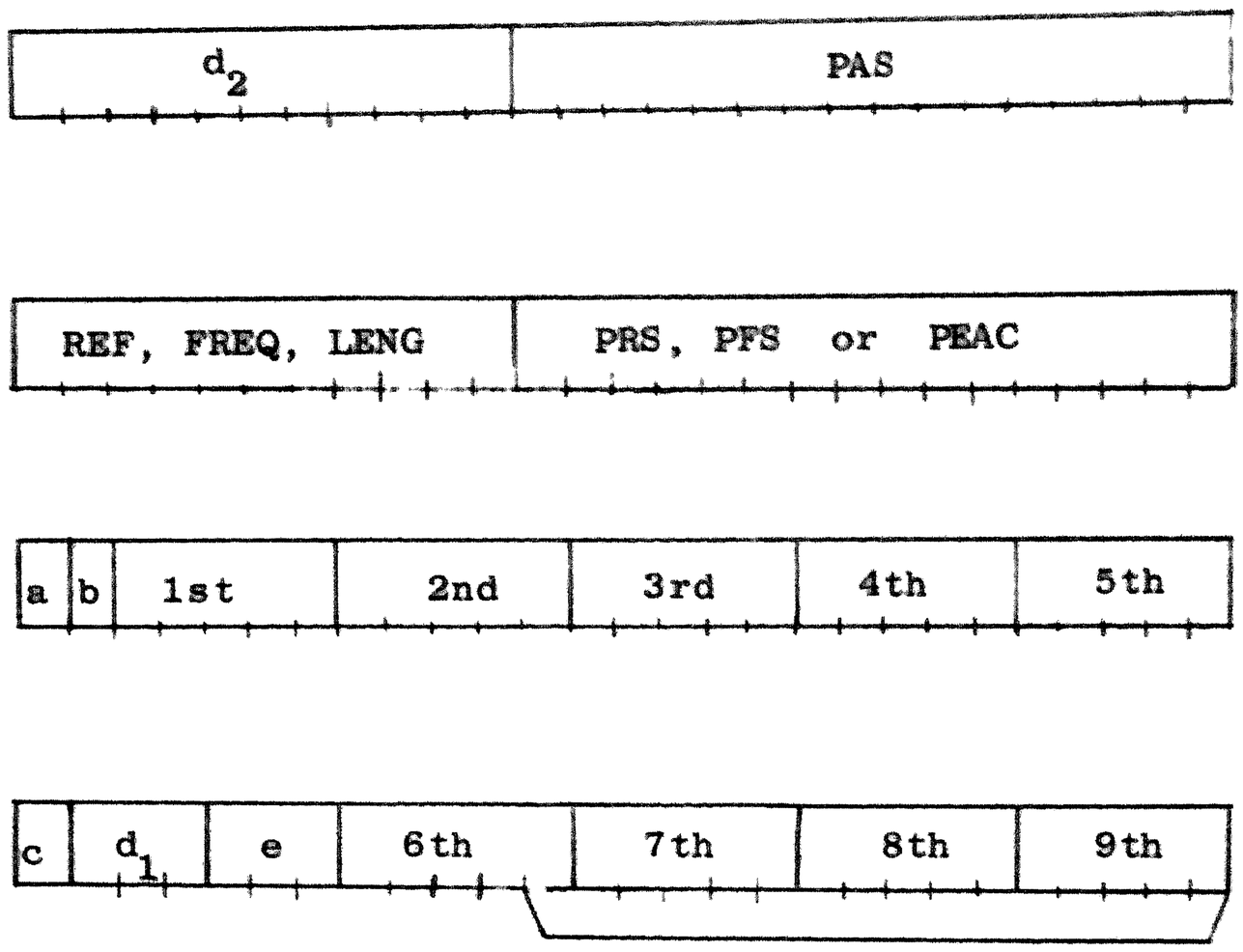

figure 2

The Word unit

a = 1 : "short-circuiting" possible

b = 1 : indication "special output"

c = 1 : word longer than 9 letters

d1 + d2 : pointer "successive unit" (14 bits)

e : homograph figure

If this unit is used as "print unit" then the second address contains the printing code (or the starting address of a printing routine) and the 4th one holds the number to be printed.

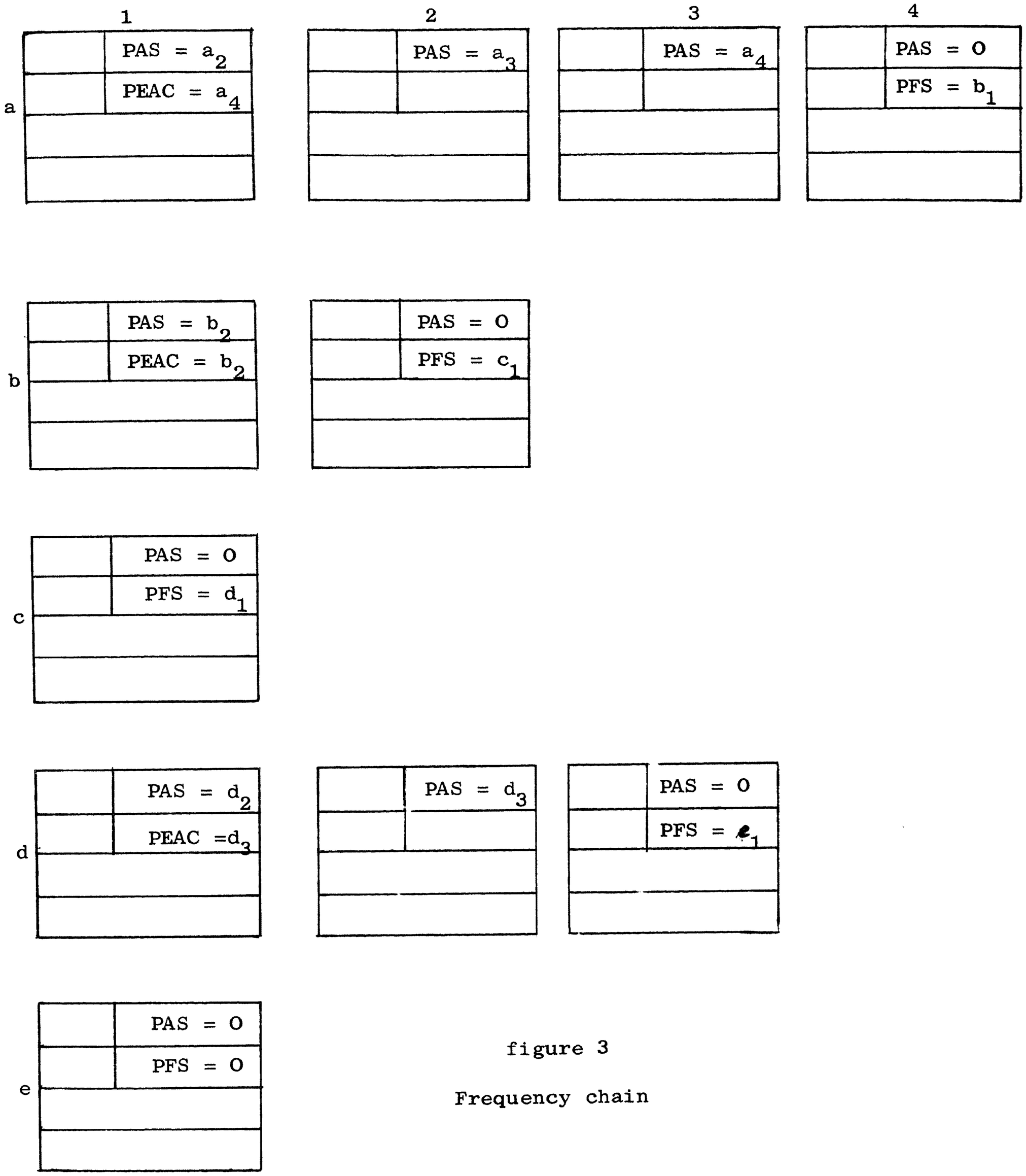

figure 3

Frequency chain

In a row the chain of words with the same frequency, in alphabetical order. In this diagram the characters a through e stand for names of chains with the same frequency and do not have any significance in connection with the alphabetizing.

Acknowledgements

I am indebted to Dr. F. de Tollenaere for his order, which enabled me to go deeply into the problem. The manner in which we collaborated was most stimulating. I owe many thanks to my colleagues of the Computation Department. I am grateful to Mr. Ch. Harmse for his help in the translation of the Dutch manuscript and the elucidations he made which will undoubtedly benefit the readability of this report. Further I wish to mention Mr. P.J.J. van de Laarschot and Mr. J. Nederkoorn for several suggestions that had much influence on the ultimate moulding of the program.

References

[1] E.W. Dijkstra: Communication with an automatic computer. Thesis, Amsterdam (1959).

[2] C.M. Popplewell and P.J. Wexler: A Word-Index to Racine's Mithridate. Faculty of Arts Manchester University Lexicological and Indexing Seminar (1960).

[3] J.A. Painter: Computer Preparation of a Poetry Concordance. Communications of the ACM vol.3, number 2, February 1960.