Formal description of a DNA oriented computer language

John L.Schroeder and Frederick R.Blattner

Department of Genetics, University of Wisconsin, Madison, WI 53706, USA

Received 12 November 1981

ABSTRACT

A computer language termed DNA* has been devised to aid in the description of DNA sequence manipulations. This was an outgrowth of a DNA sequence editor which has been implemented for a microcomputer. A formal description of the language in the BNF formalism is presented.

INTRODUCTION

A primary area of research in our laboratory has been the determination and analysis of long DNA sequences. To analyse these data we have written a number of programs for a Cromemco Z-80 based microcomputer, some of which are illustrated in Figs. 1 and 2. In this paper we would like to focus on the level of analysis that occurs prior to the running of sequence analysis programs; namely on the preparation and assembly of sequence data files. A typical example of the type of problem which we face in the laboratory is presented by the genes for the μ and δ heavy chains of immunoglobulins. The biological function of this region involves a complex series of splicings which occur at both DNA and RNA levels. A series of 15 exons exist in this DNA and as a result of alternate splicing pathways at least four different mRNAs for membrane and secreted forms of these molecules can be produced. In addition to these naturally spliced molecules, a number of different plasmid and phage clones made in the laboratory must be analysed. In order to study a particular molecule, say a clone of messenger RNA for the membrane form of μ in the PstI site of PBR322 in the reverse orientation, it is necessary to combine a number of subsections from several different sequence files. To do this it is necessary to construct

the reverse complement of a sequence, to search for a restriction site, to form a circular permutation, and to splice one sequence into another. It is difficult to use an editor oriented toward English language text to perform these tasks. A long series of commands is required even with a sophisticated conventional editor. We wanted to be able to accomplish each of these with a single operation and to construct an entire molecule with a single statement.

To accomplish this, we began to develop a DNA oriented editing program in which these concepts appeared more naturally. In writing this program we realized that DNA manipulations lent themselves to formal mathematical description and we devised a very compact notation to express them.

For this paper we have carefully reevaluated the notation, extended it, and prepared a formal description using the Backus-Naur-Form (BNF), a meta-language designed for syntactic descriptions of languages that was originally devised to define ALGOL 60 (4,5,6). The language we describe, which we call DNA*, differs in some ways from what was used in the file splicing program that inspired it. A most important difference is that DNA* employs context free constructions exclusively and has been designed so that a simple parsing program can be used to decode its sentences as they are read, without backtracking. We have also eliminated certain non-uniformities from the original notation. The language can be readily extended through the addition of functions that operate on sequences.

In the following sections we present a description of the language from the point of view of a molecular biologist user, followed by a formal description of the syntax that may be used by the computer programmer to implement or extend the language.

DNA SEQUENCE VARIABLE NAMES

In the DNA* language sequences are referred to by an assigned variable, the sequence name. The DNA sequence to which this name refers may be contained in a sequence data file, or may be a more complex structure such as sequences derived from parts of files or by joining several files. Thus a sequence name might define a sequence that includes segments from any number of primary

TABLE 1 List of DNA* Symbols

() []	interval specifications	> or	coordinate separator read right
+	sequence catenation site union arithmetic plus	<	coordinate separator read left
~	reverse complement	>>	search right
" or	enclose sequence literals	<<	search left
:	enclose site literals	#	search iteration
=	assignment	*	multiple sequence catenation
-	arithmetic minus site subtraction	%	union of site and its reverse complement
!	5' strand cutsite	?	display
		^	3' strand cutsite

sequence files. For clarity in this exposition we have employed the extension .SEQ to refer to sequence files although in the language use of the extension is optional.

The way in which a sequence name is assigned a value is by the assignment operator, = . To designate a sequence which is a sub-fragment of an existing sequence, we use a notation in which the coordinates of the ends of the sub-fragment are placed in parentheses following the sequence from which the sub-fragment is to be derived. For example:

TETGENE = PBR322.SEQ(259>1275) or

TETGENE = PBR322.SEQ(259,1275)

These statements, which mean the same thing, set up a temporary variable describing a sequence whose first base corresponds to base 259 in the PBR322 sequence file and whose 1017th base corresponds to base 1275 of PBR322. This is the region of PBR322 that codes for the tetracycline resistance gene. Although the > symbol is more graphic in indicating a direction of movement of a cursor through the sequence, many users seem to prefer the comma for indicating coordinates and thus the language treats > and , as equivalent. The numbers within parentheses

refer to an inclusively numbered DNA sequence interval. Thus on the left side of the parenthesis the coordinate specifier refers to the base after the cutsite whereas the right side specifier refers to the base before the cutsite. Square brackets can be used to designate exclusive numbering. Specifically the [designates that the coordinate is the base to the left of the cutsite and the] bracket indicates that the coordinate is the base to the right of the cutsite. By the use of mixed brackets, the discriminating user can specify coordinate intervals in any way he wants. Once a variable has been defined it can be used to define further variables. For example:

```
TETFRAG = TETGENE(500>1000)
```

would define a sequence running from PBR322 coordinates 758 to 1258. Whenever TETGENE or TETFRAG is encountered, the meaning is derived from the stored specification that describes them. No actual sequence file is created. All sequence data remains in the sequence data file PBR322.SEQ. (File creation can be accomplished, however, with the FILE command discussed below.)

By a single command it is possible to create a sequence which is the reverse complement of a defined sequence. The first way is simply to precede the sequence name with a ~ sign. Alternatively the right arrow within the coordinate specifier can be replaced with a left arrow to denote a leftward direction of reading. Thus the gene of PBR322 coding for ampicillin resistance can be defined as follow

```
AMPGENE = PBR322.SEQ(4154<3294) or
```

```
AMPGENE = ~PBR322.SEQ(3294>4154)
```

In either case the first base of AMPGENE is the complement of base 4154 of PBR322 and its 861st base is the complement of base 3294 of PBR322.

The notation we have devised also makes it easy to handle circular molecules. For example,

```
BAMPBR = PBR322.SEQ(376>375)
```

defines a permutation of the PBR322.SEQ sequence starting at 376, the BamHI site of PBR322, and proceeding around the circle ending at 375. By the same token,

```
REVBAMPBR = PBR322.SEQ(375<376)
```

is the reverse complement of the BamHI cut PBR322 molecule

obtained by proceeding around the circle in the counter-clockwise direction.

Actually, the DNA* language makes the assumption that all sequences are circular (i.e. the structures are wrapped around by calculating site positions with modular arithmetic so that if an operation reads past the end it will continue at the beginning). In this framework a linear molecule is always constructed as a sub-sequence of a circular one and there is no need to indicate on the sequence file whether the file represents a molecule that is naturally circular.

Sequences can also be literally assigned by putting the definition inside quotation marks:

```
TAIL = "GGGGG".
```

To define sequence variable names which contain data from more than one file we have created the catenation operator, denoted by +, and the repeated catenation operator, denoted by *. These specify the end to end joining of DNA sequences. For example:

```
MRNA = GENE(21>300) + GENE(370>450) + GENE(800>1000) + 200*"A"
```

splices out the intervening sequences of a gene to yield an mRNA with a 200 base pair long extension of poly A at the 3' end.

When, as in this example, several sub-sequences of a given sequence are to be joined, the source sequence name need not be repeated. Thus we could have specified:

```
MRNA = GENE(21>300) (370>450) (800>1000) +200*"A"
```

Once a sequence is defined it can be used in further assignments. For example:

```
MRNACLONE = REVBAMPBR + "GGGGG" + MRNA + ~ "GGGGG", or
```

```
MRNACLONE = REVBAMPBR + TAIL + MRNA + ~TAIL
```

specifies the insertion of MRNA into the PBR322 plasmid at the BamHI site through the use of poly G poly C tails. Note that in the second example the ~ is used to specify the use of poly C tails on the right side of the insert.

SPECIFICATION OF COORDINATES BY THE USE OF VARIABLES

The DNA* language supports the use of integer variables or simple arithmetic expressions. Numeric variables can be used within coordinate specifications if desired. There are four

predefined variables ZEND, LEND, REND and VEND used to denote the ends of sequences. These are defined as follows:

ZEND= the base before the first (Zero END)
LEND= the first base of the interval (Left END)
REND= the last base of the interval (Right END)
VEND= the base after the last (Very END)

Thus, for example

SHORTPBR = PBR322(LEND,REND - 73).

It is also possible to define other integer variables to meet specific needs, e.g.,

I = 3862

can be defined where this number will be a frequently used coordinate.

SPECIFICATION OF COORDINATES BY SEQUENCE SEARCH

The ability to specify coordinates by means of a sequence search is a powerful feature of DNA*. The purpose of the search is to permit the definition of sub-sequence endpoints without the need to deal with numerical coordinates. To specify searches the symbol >> (search right) and << (search left) are provided along with the iteration symbol #. Specifically, the operation can start at a designated position and search in a specified direction for a sequence which matches the search parameter, a site. If the nth such site is the object of the search, n# is used to indicate the operation. These may be repeated as needed in a single expression to specify a progression of searches. In general the search parameter resembles the sequence specification already described except the cursor is allowed to move back and forth a number of times through the sequence to find the starting and ending coordinates. For example in

SEQUENCE (LEND >> A << B > C << D)

one imagines a cursor which starts at the left end of SEQUENCE and moves rightward (>>) until a sequence satisfying search argument A is found. From this point a second search is initiated leftward (<<) for site B, the beginning of the desired sub-sequence. This is indicated by the >, or < symbols. The cursor proceeds to search for C and D thereby arriving at the

right end of the sub-sequence. Thus, it is possible to search for the closest B site to the left of the first A site without regard to how many other B sites occur between LEND and A.

An example of a useful search specification involves the restriction site, although more complex search arguments can be used as discussed below. For example

```
TETGENE = PBR322.(LEND>>BAMI<<MSTI-1>AVAI<<3#FNUH-13)
```

This defines the same sequence as TETGENE in an earlier example but in this case the result is obtained without the need to specify any absolute coordinates. The search starts at the left end of PBR322 and proceeds right to (the cutsite of) the first BAMI site, then left to the first MSTI site, from which 1 is subtracted bringing us to nucleotide 259, the left coordinate of TETGENE. The search then continues right to the AVAI site and left to the third FNUH site from which 13 is subtracted. This leads to position 1275, the ending coordinate of the TET gene. Searches can be restarted at any point by inserting a coordinate or coordinate variable. Thus:

```
FRAG =PBR322(LEND>>BAMI>1500<<SPH1)
```

specifies the sequence from the first BAMI site of PBR322 to the first SPH1 site to the left of 1500 in PBR322. By the use of a series of searches from rare sites to more frequent ones it is usually possible to make unique definitions of sub-sequences even if both end points are specified by frequent sites.

In the absence of an explicit starting location, DNA* assumes a rightward search beginning at LEND. For example

```
FRAG = PBR322(BAM1>1500<<SPH1)
```

produces the same result as the expression above. In the absence of an explicit starting location for the right coordinate search, the left coordinate is assumed as the starting point and the search proceeds in the direction of the single arrow by default. This leads to generally compact but unambiguous expressions e.g.:

```
FRAG = PBR322(ECR1>PVU2).
```

It should be noted that when a search direction is leftward all search arguments are automatically reverse complemented so that the site is found on the 5' to 3' strand. If this is not desired, a ~ should be placed before the site used in the search parameter.

ACAGCATGCGCCAGTCACTATGGCGTGTCTAGCGCTATATGGTTGATGCAATTTCTATGCGCACCGTTCTCGAGCACTGTCGAGCGCTTTGGCGG
TGTCTGATCGGTCTAGTGATACGCGACGAGATGCGGATATAGCGCAACTACGTTAAAGATACGCGTGGCAAGAGCTCGTGA CAGGCTGGCGAAACCGGC

P	T	TM	BOON H	MM	S
M	A	AB	AND G	PN	F
U	Q	CO	MOO A	AL	A
N	1	11	121 1	21	M

CGCCCCAGTCTGCTCGCTTCTGCTACTCTGGAGCCACTATCGACTACGGGATCATGGGACACACCCGCTCTGTGGATCTCTACGCCGAGCGCATCTGTG
CGCGGCTCAGGACGAGCGAGCGATGAACCTCGCTGATAGCTGATCGCTAGTACCGCTGGTGTGGGCAGGACACTAGGAGATGCGGCTCGGTAGCAC 400

P P S P A R F A P S S H Y L R L R D I G D H T T P F V D P L R T H R A
 R P V L L A S L L G A T I D Y A I N A H T T P V L M I L Y A G R I V
 A O S C S L R Y L E P L S T T R S M R F H P S C G S S T P D A S C

 R A C T R S A E S S F A V I S A I N A H V G T G H I R A P R N T G
 A T T A S A S A S A S A S A S A S A S A S A S A S A S A S A S A S
 G I G A R K A V O L N J S R S P S H V G T S G R R R V C N

THIS IS AN ALPHABETICAL MAP OF PRR322

[illegible]

SITE VARIABLES

Restriction sites as used in the above example actually can signify rather complex entities and this has necessitated the creation of a data type for the search argument that is more complex than the DNA sequence. This type of data is termed the site. The site consists of a list of sequences plus 5' and 3' cutsites with the entire list being referred to with a single name. Creation of such a variable is accomplished by the = sign and followed by a literal enclosed between colons (:)

```
HIN3 = :A!AGCT^T:
```

In this expression the exclamation point serves to identify the cutsite on the 5' strand and the up arrow (^) identifies the position of the cutsite on the opposite strand if not directly opposite the !. It is frequently necessary to include more than one sequence, any of which will satisfy the search, under a single name. The + is used to signify the union (merger) of an additional site, either a variable or literal, to the definition of site list variable. The - sign indicates removal of a site from a list. A simple example would be the specification of the EcoR2 site:

```
EcoR2 = :!CCAGG^: + :!CCTGG^:
```

The reverse complement operator for sites is ~. The reverse complement of ! is ^ and vice versa. Using this operation EcoR2 could be defined as:

```
EcoR2 = :!CCAGG^: + ~ :!CCAGG^:
```

The concept of combining both a site and its reverse complement is encountered frequently in site specifications and therefore a special unary operation, %, has been defined. % SITE means SITE + ~SITE; thus, still another way to define the EcoR2 site would be:

```
EcoR2 = % :!CCAGG^:
```

Figure 1 Sequence Presentation and Alphabetical Site List for PBR322.

The first program presents the positions of all restriction cutsites directly above the sequence. Below is the translation of the DNA in all 6 phases. Single letter code abbreviations are used for all amino acids and they appear directly beneath the first base of the codon.

The second program searches for all restriction sites in a sequence and presents numerical coordinates for each.

ALIGNMENT OF MU AND DELTA MEMBRANE EXONS

FIRST MU MEMBRANE EXON: \ G E
 TCTCAGAACCATCAGGGACCCCAACCTTATGCAATGCTCAGTCACCCCAAGACTGGCTTGACCCCTCCTCTCTGTGTCCCTTCTAGAGGGGGAG
 TCT AG C A C T A C T G C A A A C C A C C C T G T T A G A G G A
 TCTGTAGGGTGGAGCCTTCTCATGAGCACTAGTCTCCCTAGGCATAGTCAACACCATCCAACACTCGTGATATGGATGAGCAAAGTGACAGCTAC
 FIRST DELTA MEMBRANE EXON: \ N T I Q H B C I M D E Q S D S V

V N P E E E G F F E N L W T T A S T F I V L F L L S L F Y S T T V T L
 GTGAATCCTGAGGAGGAAGCTTTGAGAACCCTGTGGACCACTGCCTCCACCTTCATCGTCTCTCTCTGAGCCTCTTCTACAGCACCACCGTCACCC
 TG A GAGGAGGA A CCGTGG CCAC T CACCTTC T G CCTCTTCT CT A CT TCTACAG C CGTCACC
 ATGGACTTAGAGGAGGA.....GAACGGCCTGTGGCCCAATGTGCACCTTCGTGGCCCTCTTCTGCTCACACTGCTCTACAGTGGCTTCGTCACT
 N D L E E E N G L W P T H C T F V A L F L L T L L Y S G F V T F

F K /
 TGTTCAGGTAGTA.....
 T TCAAGGTAG
 TCAATCAAGGTAGCTCCTCTACCACTCCTTCTGAACCTTCTGGGCACATGAAGATACCTTCCAGGGGCATTTTGAAACCCATCGGTACTTCTGGAG
 I K /

.....TGGTTGTGGGGCTGAGGACACAGGGCTGGGACAGGGAGTCACCACTCTCACTGCCTCTACCTCTACTCCCTACAAAGTGA
 T T T G G G G C G C G C G G G A C C A C G C T T C C T T C A C A A G
 TGGCTAGGGAATGCACAAGTAATGGGAGGCATGGTCTCAGCTTCTGGGCCCTATGTAGCTCCAGGACTTTCCTGTTCCTCAGACCAAGCC

MU CYTOPLASMIC EXON: V / K
 CAGCAATTCACACTGTCTGTTCACCTGCAGGTGAAATGACTCTCAGCATGGAAGGACAGCAGAGACCAAGAGATCCTCCACAGGGGACACTACCTCTGG
 AT ACT TC C GCAGGTGAA T C CA A C GCA AC AGA A C C A T C
 TCTGTAT...GACTTCACGGCTCTCTGGCAGGTGAAGTAGACAGGACAGCAGAATCCTGCAACTACAGAGAAAGTGTCTTCCCTCAACATGAAGCCAA
 DELTA CYTOPLASMIC EXON: / V K

THIS IS A MAXAM GILBERT SEQUENCING STRATEGY SEARCH OF B:PBR322.LST
 IN REGION FROM BBV1 SITE AT 613 TO ASU1 SITE AT 886

THE SEARCH IS FOR FRAGMENTS THAT CAN BE END LABELED
 WITHIN 50 OF THE DESIRED AREA, RECUT AND RUN ON A GEL AS A
 FRAGMENT SMALLER THAN 500 AND RUNNING NO CLOSER
 THAN 10 PERCENT TO ANY OTHER LABELED FRAGMENT

HOT END (DIR) / OTHER END	DIST TO SEQ /	NEXT BELO /	FRAG TO SEQ	/NEXT ABOVE/
*HGIA R SAC3	0	0	309	4363
*SAC3 L HGIA	13	247	309	357
*HGIA R HAE1	13	0	329	4363
*HAE1 L HGIA	33	292	329	1001
*HGIA R BGL1	23	310	344	466
*BGL1 L HGIA	48	234	344	466
*HGIA R GDI2	23	291	348	498
*HGIA R XNA3	23	310	348	498
*HGIA R NRU1	23	310	380	498
*HGIA R ECR2	23	310	467	826
*SPH1 R SAC3	48	0	334	642
*SAC3 L SPH1	48	230	334	373
*SPH1 R HAE1	48	0	354	1975
*HAE1 L SPH1	33	72	354	397
*SPH1 R BGL1	48	0	369	1440
*BGL1 L SPH1	48	234	369	1440
*SPH1 R GDI2	48	35	373	0
*SPH1 R XNA3	48	0	373	3989
*SPH1 R NRU1	48	0	405	3957
*SPH1 R ECR2	48	436	492	0
*SAC3 L ACY1	48	250	351	4363
*HAE1 L ACY1	33	240	371	452
*BGL1 L ACY1	33	234	386	2282
*SAC3 L NAR1	33	250	351	4363
*HAE1 L NAR1	33	240	371	452
*BGL1 L NAR1	48	234	386	1289
*SAC3 L CAU2	48	322	365	628
*HAE1 L CAU2	48	324	385	4363
*BGL1 L CAU2	48	234	400	633
*SAC3 L GDI2	48	320	369	416
*HAE1 L GDI2	48	57	389	4363
*BGL1 L GDI2	48	268	404	2319
*SAC3 L BGA1	48	230	411	589
*HAE1 L BGA1	48	72	431	4363
*BGL1 L BGA1	48	294	446	714
*SAC3 L HPH1	13	343	454	950
*HAE1 L HPH1	13	385	474	4363
*BGL1 L HPH1	48	365	489	0

THE STRATEGY SEARCH IS DONE GOOD LUCK

To facilitate the specification of ambiguous nucleotides in a site specification curly brackets or X's can be used. This results in the addition to the sitelist of all possible combinations. Thus the specification:

```
Hgia = :G^(AT)GC(AT)!C:
```

generates four sequences which are all added to the list of sites specified by that variable name. The EcoK site would be specified as:

```
EcoK = % :!TGAXXXXXXXXXTGCT^:
```

The site specification is by no means limited to the restriction site or to symmetrical sites. For example one might define poly(A) addition sites as follows:

```
ASITE = :AATAAAXXXXXXXXXXXXXXXXXX!+:AATTAAAXXXXXXXXXXXXXXXXXX!:
```

This reflects the fact that in eukaryotic mRNA either AATAAA or AATTAAA is usually found located about 15 nucleotides 5' to the position at which poly(A) may be added to eukaryotic mRNA.

It is sometimes useful to define a site in terms of a sequence. This may be done by the SITEOF function which allows a pair of cutsites to be associated with a sequence. The function has the form

SITEOF (SEQUENCENAME or a "literal", location of 1, location of ^). Thus the example above could have been written:

```
ASITE = SITEOF("AATAAA"+15*"X",21,21) +  
SITEOF("AATTAAA"+15*"X",22,22).
```

It is also useful to determine the existence of or to find the position of a site for some trial sequence. For this purpose, the function

POS (any search expression, any DNA sequence expression)
returns the specified coordinate for the DNA sequence expressed

Figure 2 Output of Alignment Program and Strategy Search.

The output of the first program aligns two genes, μ and δ , sharing evolutionary homology with gaps placed to maximize sequence agreement. Bases which agree are repeated between the sequence.

The second program calculates all possible sites for end-labelling and recutting so as to yield fragments on a gel which are within a specified size range, resolved from often labelled fragments and labelled at a site within a specified distance from the region it is desired to sequence.

as the second parameter, if it exists. Therefore

```
I = POS(H1N3>>ECR1, PBR322(100,500) (700,900))
```

```
DISPLAY I
```

tests the existence of an Eco R1 site following a Hin3 site in PBR322 within the specified ranges.

DISPLAY OF VARIABLES

The display of variables or expressions may be accomplished with the DISPLAY command (or ? as a shortened form).

```
DISPLAY TETGENE or ? TETGENE
```

presents a list of sub-sequences and their endpoints. The display of a site variable presents a list of literal sequences associated with its definition. Numeric variables or expressions may be displayed as a decimal value.

A list of names of all currently defined sites may be produced with the command 'DISPLAY SITES'. Similarly, the names of DNA variables may be viewed with 'DISPLAY DNAS', and current file names may be viewed with 'DISPLAY FILES'.

PERMANENT STORAGE OF VARIABLES

The assignment of sequence variable names and site variable names discussed so far leads to the creation of temporary variables. To create a file containing a sequence specified by a DNA sequence variable, the FILE command is used.

```
FILE TETGENE AS TET.SEQ.
```

This leads to the creation of a sequence file under the name TET.SEQ, corresponding to the variable TETGENE. The command UNFILE can be used to eliminate any file created by FILE. The same commands, FILE and UNFILE, are used to store and remove site definitions in the restriction site list. The result is the storage or erasure of the definition in this data base rather than the construction of an independent file for each site.

AN EXAMPLE PROGRAM

The following is an example of a DNA* program which uses DNA sequence files that exist in our file library. The third line of this example solves the problem presented in the introduction.

```
MUSECRETED = VBCL(123>168)(251>567) + JHREGION(764>809)
      + MU6(102>416)(527>865)(1144>1461)(1569>2090)
MUMEMBRANE = MUSECRETED(1>VEND-89) + MUMEM(155>270)(389>670)
PSTCLONE=PBR322(PST1<PST1)+ 20*"G" +MUMEMBRANE+ 200*"A" + 20*"C"
FILE MUSECRETED AS MUSEC
FILE MUMEMBRANE AS MUMEM
FILE PSTCLONE
```

To accomplish the same thing with the TECO text editor would require more than 60 command lines.

INTERFACING DNA* TO OTHER PROGRAMS

Once a sequence has been filed, any program designed to operate on sequence files can be used to analyze that sequence. This is the simplest way to allow the products of DNA* operations to be used relative to programs which do not utilize this language. The language is readily extensible by the addition of functions which would allow calling user designed sequence analysis programs.

FORMAL DEFINITION

The formal definition of DNA* appears in Fig. 3. The BNF notation that we have used to represent the syntax of DNA* is the most commonly accepted method of describing syntax. The symbols ::= and : are meta-symbols of the BNF notation. Sentences in BNF are called productions and are constructed from the symbols of the language to be defined and meta-symbols. The symbol to the left of the ::= names the sequence of symbols to the right. Symbols separated by : are alternative definitions. We have also adopted the symbol ε as an alternative definition for symbols that may be absent.

The set of productions presented in Fig. 3 reflect some aspects of the structure of the translating program. The productions are context free and right recursive. These two characteristics make it possible for the parsing algorithm to determine which production to use to correctly recognize a sentence without having to retrace its steps. As a result, a goal-oriented or top-down parsing algorithm may be used. It is

```

(sentence) ::= (statement) (newline)
(statement) ::= (empty) : (command) : (assignment)
(command) ::= (file command) : (unfile command) : (display command)
(file command) ::= file (storage object) (newname part)
(storage object) ::= (DNA sequence) : (site)
(newname part) ::= (empty) : as (identifier)

(unfile command) ::= unfile (removed object)
(removed object) ::= (DNA sequence identifier) : (site identifier)

(display command) ::= (display key) (displayed object)
(display key) ::= display : ?
(displayed object) ::= sites : dnas : files : (object)
(object) ::= (DNA sequence) : (site) : (expression)

(assignment) ::= (identifier) = (object)

(DNA sequence) ::= (DNA expression) (catenated part)
(catenated part) ::= (empty) : + (DNA expression) (catenated part)
(DNA expression) ::= (DNA term) : (n-value) * (DNA factor)
(DNA factor) ::= (DNA term) : ( (DNA sequence) )
(DNA term) ::= (DNA literal) : (DNA identifier) (sub-part) : (complemented DNA object)
(DNA literal) ::= (quote) (sequence) (quote)
(sequence) ::= (empty) : (base) (sequence)
(base) ::= (nucleotide) : { (ambiguous nucleotide) }
(ambiguous nucleotide) ::= (empty) : (base) (ambiguous nucleotide)
(quote) ::= ' : "
(nucleotide) ::= A : C : G : T : X
(DNA identifier) ::= (identifier)
(sub-part) ::= (empty) : (open) (limiting part) (close) (sub-part)
(open) ::= ( : [
(close) ::= ] : )
(complemented DNA object) ::= ~ (DNA factor)
(limiting part) ::= (search expression) (separator) (search expression)
(separator) ::= , : ; : {
(search expression) ::= (search term) (offset part)
(offset part) ::= (empty) : (sign) (search term) (offset part)
(search term) ::= (positional expression) (search part)
(search part) ::= (empty) : (search direction) (positional expression) (search part)
(search direction) ::= ( ( : ) )
(positional expression) ::= (site) : (term)
(term) ::= (n-value) (repeated part)
(repeated part) ::= (empty) : # (search factor)
(search factor) ::= (site) : ( (search expression) )

(site) ::= (site expression) (union part)
(union part) ::= (empty) : (union) (site expression) (union part)
(union) ::= + : -
(site expression) ::= (site term) : % (site term)
(site term) ::= (site literal) : (site identifier) : (complemented site object)
: (site conversion expression)
(site literal) ::= : (site sequence) :
(site sequence) ::= (empty) : (site element) (site sequence)
(site element) ::= ! : ^ : (base)
(site identifier) ::= (identifier)
(complemented site object) ::= ~ (site factor)
(site factor) ::= (site term) : ( (site) )
(site conversion expression) ::= sizeof ( (DNA sequence) (cut specification) )
(cut specification) ::= (empty) : , (n-value) (3' cut)
(3' cut) ::= (empty) : , (n-value)

(expression) ::= (n-value) (arithmetic part) : (sign) (n-value) (arithmetic part)
(arithmetic part) ::= (empty) : (sign) (n-value) (arithmetic part)
(sign) ::= + : -
(n-value) ::= (unsigned constant) : (n-value identifier)
: (n-value conversion expression) : ( (expression) )
(n-value identifier) ::= (identifier)
(n-value conversion expression) ::= pos ( (search expression) , (DNA sequence) )

(identifier) ::= (letter) (more identifier)
(more identifier) ::= (empty) : (letter) (more identifier) : (digit) (more identifier)
(unsigned constant) ::= (digit) (more constant)
(more constant) ::= (empty) : (digit) (more constant)
(empty) ::= ε

```

relatively straightforward to construct this type of parsing algorithm from a list of productions. In fact, programs to construct parsing programs of this type are currently in use (7). It is worthy of note that each production may be directly associated with some fraction of the underlying interpretation of the sentence being read. Therefore, these productions not only determine what is recognized as correct, but provide a structure that is of value in the recognition of meaning within sentences.

CONCLUSION

The language we have devised is simple as computer languages go. It has three data types, (sequence, site and integer), operations for combining each of them (sequence catenation (+), site union (+ and -) and arithmetic (+ and -)), a unary complementation operation for sites and sequence (~), and a system of operations for decomposing sequences into sub-sequences. The language uses the symbols shown in Table 1.

What has been presented here is a core language. Even with this limited capability it has been very useful in the design of a sequence editor. Obviously DNA* can be extended in a number of directions to make it more versatile. More elaborate display functions that provide the kind of information in fig. 1 could be added to make the language more internally complete. As we have stated, many DNA analysis programs exist in a form that requires the sequence file as input data. Such programs could be called as functions within the context of DNA*, making the file an unnecessary step in the chain of programs. Of course, to accomplish that, a database for DNA* variables would be of considerable convenience, along with syntax for describing classifications of variables. In such a form, DNA* could be the

Figure 3 The Syntax of DNA* Language Expressed in BNF Formalism.

ϵ represents the empty set. <newline> represents the carriage return character, but it may be replaced with ; for compatibility with algorithmic languages. The last group of productions which define the construction of identifiers and constants are normally performed by a lexical scanner rather than the parsing program. Note that the symbols (and) which are normally symbols of extended BNF are part of DNA* and are used nowhere in the document as part of BNF.

core of an algorithmic language and thus support any series of sequence analysis tasks without supervision. Nonetheless, the core language is valuable enough to be presented as we have defined it, without the extensions.

ACKNOWLEDGEMENTS

We gratefully acknowledge Donna L. Daniels, Thomas R. Virgilio, Julia E. Richards, and Oliver Smithies for helpful discussions, improvements upon the original ideas, and critical reading of the manuscript. We also wish to thank Pat Parish for patiently typing the manuscript. This work was supported by grant GM 28252 from the National Institute of General Medical Sciences. This is paper 2547 from the laboratory of Genetics, University of Wisconsin.

REFERENCES

1. DeWet, J.R., Daniels, D.L., Schroeder, J.L., Williams, B.G., Denniston-Thompson, K., Moore, D.D., and Blattner, F.R. (1980) *J. of Virology* 33:1, 401-410.
2. Daniels, D.L., Schroeder, J.L., Au-Yeung, P., and Blattner, F.R. (1980) in *Genetic Maps*, Steven J. O'Brien, ed., Vol. 1, pp. 4-15.
3. Goldberg, G.I., Vanin, E.F., Zrolka, A.M., and Blattner, F.R. (1981) *Gene*, 15, 33-42.
4. Knuth, D.E., (1971) *Top-Down Syntax Analysis*, *Acta Information*, 1, no. 2, pp. 79-110.
5. Naur, P., ed., (1963) *Report on the Algorithmic Lang. ALGOL 60*, ACM, 6, no. 1, pp. 1-17.
6. Lewis, P.M., Stearns, R.E. (1968) *Syntax Directed Transduction*, 15:3, pp. 465-488.
7. Johnson, S.C. (1975) *Yacc: Yet Another Compiler Compiler*, *Computing Science Technical Report 32*, Bell Labs, Murray Hill, N.J. 07974.
8. Sutcliffe, G. (1978) *Nucleotide Sequence of pBR322*. *Cold Spring Symp. Quant. Biol.* 43; 77-90.