The design of synthetic genes

Scott R.Presnell and Steven A.Benner

Laboratory for Organic Chemistry, ETH–Zentrum, Universtätstrasse 16, CH-8092 Zurich, Switzerland

Received August 17, 1987; Revised and Accepted November 9, 1987

ABSTRACT

Computer programs are described that aid in the design of synthetic genes coding for proteins that are targets of a research program in site directed mutagenesis. These programs "reverse-translate" protein sequences into general nucleic acid sequences (those where codons have not yet been selected), map restriction sites into general DNA sequences, identify points in the synthetic gene where unique restriction sites can be introduced, and assist in the design of genes coding for hybrids and evolutionary intermediates between homologous proteins. Application of these programs therefore facilitates the use of modular mutagenesis to create variants of proteins, and the implementation of evolutionary guidance as a strategy for selecting mutants.

INTRODUCTION

Interest in the application of "site-directed mutagenesis" to the study of the relationship between structure and function in proteins is exploding[1]. The technique offers much promise, including the design of "tailor-made" enzymes for chemical reactions[2]. Much of this promise remains unrealized, undoubtedly because of the complexity of the problem being addressed and the limited rationales used to select mutations[3]. New insights into the workings of proteins will certainly be gained using these techniques, but progress will depend both on creative new ideas and much hard work.

To begin a research program in this area, genes coding for the target protein must be obtained. With recent advances in methods for automated synthesis of nucleic acids, such genes are now more readily obtained by synthesis than by cloning, provided that they are moderate in size and the sequence of the target protein is known.

Synthesis has several advantages when compared to cloning genetic material from natural sources. Codons can be chosen in such a way as to place unique restriction endonuclease recognition sites at intervals in the gene. These restriction sites allow for the rapid construction of genes coding for altered proteins through the replacement of nucleic acid modules (modular mutagenesis). Codons may also be chosen to optimize the level of heterologous expression of the gene[4]. While there is not yet an unambiguous demonstration of the importance of codon usage for high expression, a synthetic gene gives the biochemist full control over this parameter as well.

Finally, a synthetic gene can be designed to facilitate cloning and expression. Provisions for the placement of the synthetic gene with respect to promoters, operators, ribosome binding sites, and transcription terminators can all be included in the design for the synthetic gene.

Surprisingly, synthetic genes are rarely used in site-directed mutagenesis. In some cases, the cloning of the gene preceded research using site-directed mutagenesis[5]. In other cases, the gene being studied is from a bacterial source, where cloning is rather simple[6]. Even when genes are synthesized, the full advantages of gene design are only rarely exploited. In the gene coding for interferon, the first synthetic gene for a large protein[7], only two restriction sites were introduced. Examples of synthetic genes where strategic codon selection was only partially utilized are found in references (8-16).

A gene coding for bovine pancreatic ribonuclease (RNase) was the first to be synthesized explicitly to support studies using site-directed mutagenesis techniques and the first to incorporate by design the features outlined above[17]. Restriction sites were placed at approximately uniform intervals throughout the gene. Codons not constrained by a need for a restriction site were then chosen to maximize efficiency of expression in *E. coli*. Finally, restriction sites were introduced at the termini of the synthetic gene to make it a modular cloning unit.

Others have followed this strategy since. Excellent examples include the gene for the human complement fragment C5[18], a synthetic gene for bovine rhodopsin[19], and a gene coding for mammalian cytochrome b5[20].

Designing a gene with appropriately selected codons is a complex task, especially if done manually. Should evolutionary guidance be used as a strategy for selecting mutations to construct[21], the task becomes still more complicated. In the evolutionary guidance strategy, the sequences of two or more homologous proteins with different behaviors are compared, and the comparison guides the selection of mutants to test hypotheses relating structure and behavior in these proteins. Thus, if the genes for several homologous proteins are designed to share key restriction sites, mutagenesis experiments can be greatly facilitated.

We report here a collection of programming tools that can assist the biochemist in the construction of genes with a variety of DNA structural goals in mind.

THE GENE DESIGN TOOLS

Theory

Useful gene design tools should allow the user to extract from the target protein the maximum amount of information to assist in the *de novo* design of a gene coding for a protein. At the minimum, this includes (a) the conversion (by reverse translation) of the protein sequence into a general DNA, i.e., one where the codon choices have not yet been fixed in the structure of the gene; (b) identification of potential restriction sites in this general DNA sequence; (c) assisting in the identification of potentially optimal distributions of these sites.

The collection of programs described here can assist in the design of a complete, optimal gene from any known protein sequence with a range of user interaction. These programs permit the intelligent manipulation of the available data, a drafting board on the level of a single gene.

Implementation

The core group of tools is written in a dialect of Lisp computer language supported by FRANZ Inc. The Lisp language allows the internal representation of the data to be similar to the raw data. The programs were written under the UNIX program development environment on a VAX 11/785. Each program operates with

options selected from the command line. The one exception is *ged* (the main program) which supports a menu of options. Support programs were written in the 'C' computer language.

Data Representation

The internal data structures of these programs are in the form of lists, a natural representation for linear protein and nucleic acid sequences. A generalized DNA structure representation is handled by sublists, which contain the information describing the way the data at the top level are not fully specific (see below).

Modifications in the gene structure that are contemplated (e.g., restriction sites that will be introduced or removed) are also stored in lists. These lists contain the information necessary to effect the intended modification, the functions to be executed, the codons they will affect, and the priority of the operation. In this way, the modifications to be made reside in data structures that are similar to 'instances' in object oriented programming environments.

Finally, within the main program there is a list (in the form of a tree of function calls) of all the commands that have been executed. This allows the user to view the decision process.

Internal Data Base of Restriction Endonucleases

The data base containing restriction endonuclease recognition sites is a frame based knowledge representation system that includes inheritance. In this case, inheritance is an efficient mechanism to allow common information to be shared between data objects. The data base can grow by interaction with the user. If the program encounters a situation where it lacks necessary data, it will query the user, and store the information obtained as a result of the query in the data base.

THE PROGRAMS

The gene design tools consist of the four programs, *map*, *zuruck*, *diffmap*, and *ged*, described below.

Map

Map, the restriction site mapping utility, uses the following algorithm.

1) read in DNA/RNA bases.
   2) for each restriction endonuclease
      3) starting at base 1
         4) Determine the specific sequences from the general sequence necessary to completely test for the occurrence of the restriction endonuclease site.
            5) for each specific sequence
               6) Do any of the DNA bases exclude the corresponding base in the restriction site?
                  no : we have a site, pop to 8)
                  yes: continue
            7) repeat
         8) next base
   9) next endonuclease

The natural mapping of codons onto amino acids creates a complication: there is no direct way to represent the amino acids that use six codons. To reconcile this complication the DNA sequence is stored as multi-level lists. For example, "(C G T ((T A) (C G) (N Y)) G C N)" is the DNA sequence corresponding to

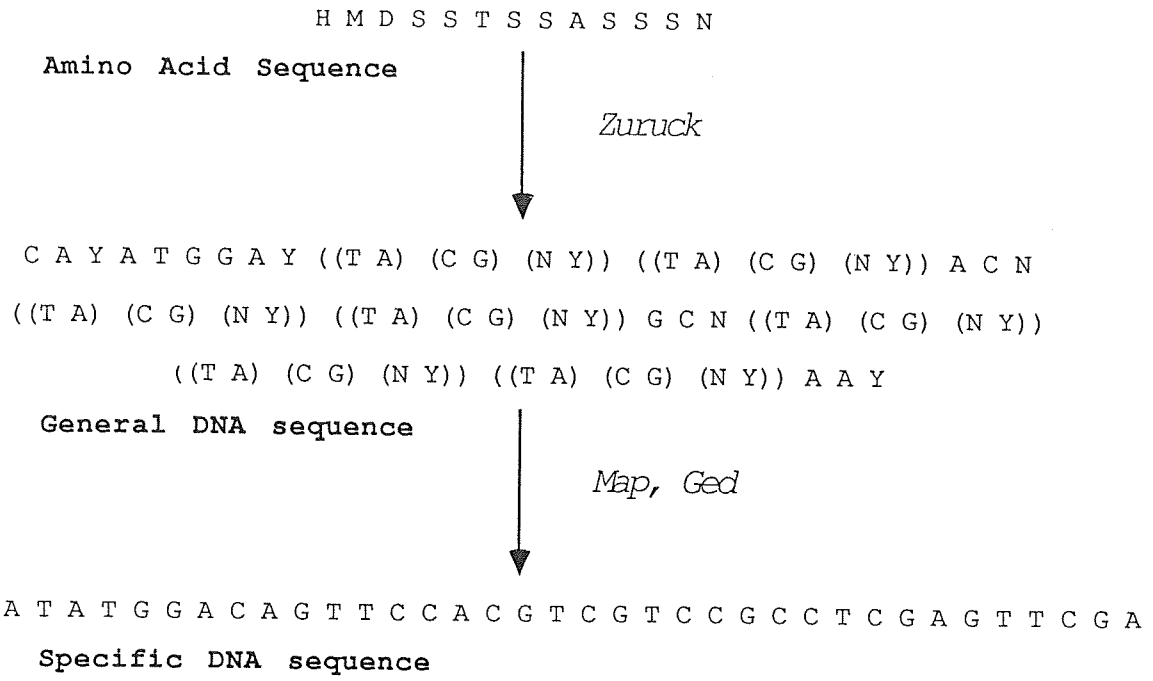

Figure 1: The flow of information in the construction of a modular mutagenesis segment.

the peptide sequence Cys-Ser-Gly. The internal list is still the length of a codon but two alternatives are available within the list structure. To access the internal sublists, the Lisp mapping functions are used. For example:

mapcar 'extract-first ((T A) (C G) (N Y))

yields (T C N). Using these techniques, *map* automatically generates and tests the different sequences possible when there are adjacent six-fold degenerate codons in the sequence.

Zuruck

Zuruck simply translates a specific protein sequence into a generalized DNA sequence using a table. This table can be altered by the user to contain 'composite residues' where codons are selected from two real amino acids. Thus ((A C) (C C) (N N)) might represent a single codon that specified either threonine or proline in a general DNA sequence. This makes the generalization of a DNA sequence flexible enough to display ambiguities that are introduced at the amino acid level.

Diffmap

Diffmap allows the comparison of partially homologous DNA or protein sequences based on their restriction site mapping. The output of this program is a list of potential restriction sites common to genes coding for two protein sequences, and a list of the restriction sites found in the primary sequence but not found in the secondary sequence.

The algorithm employed simply compares the restriction map data of the two sequences using an alignment as an offset. *Diffmap* does not determine alignments of the sequences to be compared, this information must be entered by the user. The main purpose to *diffmap* is determining sites that would allow 'gene swapping' in the construction of hybrid genes and proteins.

a

Restricion sites in rm22

Enzyme	Sequence	Sites at:
AatII	((G A C G T C))	15
AccI	((G T M K A C))	12
AhaII	((G R C G Y C))	15, 24
AluI	((A G C T))	10, 19, 24, 28, 31
AvaI	((C Y C G R G))	9, 18, 27
BanII	((G R G C Y C))	18, 27
BbvI	((G C W G C))	11, 20, 29, 32
Bsp1286	((G D G C H C))	12, 18, 27
DdeI	((C T N A G))	21
EaeI	((Y G G C C R))	23
Fnu4HI	((G C N G C))	11, 20, 29, 32
FnuDII	((C G C G))	24
FokI	((C A T C C) (G G A T G))	11, 17, 20, 26, 29, 32
HaeII	((R G C G C Y))	22
HaeIII	((G G C C))	24
HgaI	((G A C G C) (G C G T C))	25
HgiAI	((G W G C W C))	12, 18, 27
HhaI	((G C G C))	23
HincII	((G T Y R A C))	12
HinfI	((G A N T C))	7
MboII	((G A A G A) (T C T T C))	10, 19, 28, 31
MnlI	((C C T C) (G A G G))	11, 17, 20, 26, 29, 32
NdeI	((C A T A T G))	1
NheI	((G C T A G C))	25
PaeR7I	((C T C G A G))	9, 18, 27
PleI	((G A G T C) (G A C T C))	7
SacI	((G A G C T C))	18, 27
SalI	((G T C G A C))	12
ScaI	((A G T A C T))	13
SfaNI	((G C A T C) (G A T G C))	25
SpeI	((A C T A G T))	16
TaqI	((T C G A))	10, 13, 19, 28, 31, 34
Tth111I	((G A C N N N G T C))	15

b

Restricion sites in rm22_1

Enzyme	Sequence	Sites at:
AvaI	((C Y C G R G))	27
MnlI	((C C T C) (G A G G))	26
NdeI	((C A T A T G))	1
PaeR7I	((C T C G A G))	27
TaqI	((T C G A))	28, 34

Figure 2a:Restriction sites available in the segment before the design process. b: Restriction sites in the segment after the design process.

Ged

Ged's main function is to perform three design tasks: 1) determining the optimal placement of the restriction sites available; 2) incorporating these restriction sites into the gene while leaving the gene in the most generalized state possible; 3) specifying sequences for codons that had not been 'fixed' during previous manipulations. These functions are organized into 'passes' which can be applied to the designed gene as a group, or singularly, at the users discretion.

```
                Welcome to the Computer Aided Gene Design System

          Copyright (c) 1987 Scott R. Presnell. All rights reserved.

     Options:

     <a>,<apply> algorithm to a seq.,    <c>,<core>: save a core dump,
     <d>,<design>: automatic design,     <e>,<exec>ute a specific pass,
     <f>,<fork> a new process            <h>,<help>: print this menu,
     <i>,<inquire>: query the system,    <l>,start a <log> file,
     <r>,<read> in a sequence,           <p>,<print> codons to the screen,
     <q>,<quit> to UNIX;                 <m>,<manual> control,
     <s>,<save> DNA to an ASCII file,    <w>,<write> codons to a dna file,
     <v>,<ver>sion number,               <x>,<exit> to lisp,
     <?>,<help>,                         <!>,start a <shell>,
     <fv>,set <verbose> flag,            <fq>,set <quiet> flag,
     <fd>,set <debug> flag,

     ged> design

     Which algorithm (I, II, III): III
     Please enter the sequence to be used: angio
     codon1 codon2
     [... reading in of restriction data and DNA sequence ...]
     codon121 codon122 codon123
     Parsing Done
     Pass 1
     [no action]
     Pass 2
     (addsite (BglI 51))  succeeded.
     (removesite (BglI 92)) succeeded.
     (addsite (Tth111I 95)) succeeded.
     [... attempts at addition or removal of restriction sites ...]
     (removesite (MnlI 96)) failed.
     (removesite (MnlI 93)) failed.
     Pass 3
     (C A R) => (C A G)
     [... selection of codons ...]
     (C C W) => (C C A)

     ged> write
     Please enter the name for the new sequence: nangio
     ged> quit
```

Figure 3: The log of a session with the program 'ged': the design of angiogenin.

Ged supplies three different algorithms for determining the optimal placement of the restriction sites available. The first algorithm selects restriction sites to be incorporated according to the following hierarchy; 1) user assigned priority, 2) absolute length of recognition sequence, and 3) type of overhang left by the restriction endonuclease (blunt or not). Ged may be instructed to disregard the quality of bluntness if so desired.

The second algorithm is different from the first in that it uses the specificity of the restriction endonuclease rather than the length as a basis for selection. Specificity is determined by evaluating each restriction endonuclease on the degeneracy allowed at each base of the recognition sequence. This eliminates the bias of the first algorithm towards restriction recognition sequences which contain spacer regions that do not confer specificity to the enzyme (sequences of this type are also known as interrupted palindromes).

```
CAG GAT AAC TCT AGA TAC ACT CAT TTT CTG ACG CAG CAT TAC GAC
GTC CTA TTG AGA TCT ATG TGA GTA AAA GAC TGC GTC GTA ATG CAG
            XbaI                          BbvI, HgaI   AhaII

GCC AAG CCA CAA GGG CGT GAC GAT CGT TAC TGT GAG TCT ATC ATG
CGG TTC GGT GTT CCC GCA CTG CTA GCA ATG ACA CTC AGA TAG TAC
          BglI          Pvu I             Hinf I

CGC CGA CGT GGT CTG ACT AGT CCG TGT AAA GAT ATC AAC ACT TTT
GCG GCT GCA CCA GAC TGA TCA GGC ACA TTT CTA TAG TTG TGA AAA
Hha I Tth111 I    Spe I               Eco RV

ATC CAT GGT AAC AAG CGC TCT ATT AAA GCG ATC TGT GAA AAC AAA
TAG GTA CCA TTG TTC GCG AGA TAA TTT CGC TAG ACA CTT TTG TTT
  Nco I         Hae II, Hha I

AAC GGT AAT CCT CAC CGT GAA AAC CTA AGG ATT TCC AAG AGC TCT
TTG CCA TTA GGA GTG GCA CTT TTG GAT TCC TAA AGG TTC TCG AGA
               Hph I        Dde I, Mst II   Sac I, Ban II

TTC CAG GTT ACC ACC TGC AAA CTA CAC GGT GGG TCC CCG TGG CCA
AAG GTC CAA TGG TGG ACG TTT GAT GTG CCA CCC AGG GGC ACC GGT
Bst NI, Bst EII  BspM I               Eco 0109   Hae III

CCG TGT CAG TAT CGT GCA ACC GCG GGC TTT CGT AAC GTT GTT GTG
GGC ACA GTC ATA GCA CGT TGG CGC CCG AAA GCA TTG CAA CAA CAC
                          Sac II

GCA TGC GAA AAC GGC TTG CCA GTG CAC CTT GAT CAA TCG ATT TTC
CGT ACG CTT TTG CCG AAC GGT CAC GTG GAA CTA GTT AGC TAA AAG
Sph I                     ApaL I          Cla I

CGT CGA CCA
GCA GCT GGT
Acc I, Sal I
```

Figure 4: Restriction map of a designed angiogenin gene.

Both of these algorithms operate on the principle that all sites that recognize six or more bases should be added if possible, and all sites that are four bases long (the currently observed minimum length) should be removed. Sites with a length of five bases are allowed to 'fall where they may'.

Usually, the first algorithm is used to design or optimize oligos that are used in M13 or modular mutagenesis. The second alogrithm is used when a substantial section of a gene is being reorganized or redesigned - the sequence is not long enough to break into regions (see below), however there are enough restriction sites that careful site selection is prudent. Surprisingly, these simple algorithms can create useful designs. This is probably because short sequences have only a few potential restriction sites, usually unique, and any introduced restriction sites are welcome.

The third algorithm combines the idea of specificity with the fact that medium or long DNA sequences may contain more than one location to place a given restriction site. Designing an entire gene from scratch is done best with the aid of this algorithm. The DNA sequence to be designed is partitioned into regions, and the number of possible restriction sites for that region is determined and stored in the distribution table. Regions are defined as a fixed length, controllable by the user, usually about 25 bases long. The algorithm then considers the list of potential restriction sites using the hierarchy, 1) user assigned priority, and 2) specificity of the restriction endonuclease. For each enzyme the following rules are then examined:

1. If there is only one placement possible for this restriction endonuclease then try to use it. Update the distribution table to reflect the introduction of this site. This lowers the effective probability that the next restriction site will be placed in this segment.
2. If there is more than one placement possible for this restriction endonuclease, consult the distribution table. Attempt to introduce a site in the region that has the lowest number of possible restriction sites. Update the distribution table to reflect the introduction of this site. Mark the other possible placement sites for this enzyme for destruction and update the distribution table to reflect the loss of potential restriction sites.

Manual control allows the user to introduce his own hierarchy of requests that will be executed before, or along with, the machine generated requests, through the use of the priority setting. In this way, the user may select his own preferences for restriction site placement working with the programs optimal site distribution or independently.

DISCUSSION

These programs have many of the properties outlined above as useful for designing genes to be synthesized, supporting a research program in site directed mutagenesis, and assisting in implementation of modular mutagenesis as a method for constructing mutant proteins. These tools have been used in our laboratories to design mutants of bovine ribonuclease under the evolutionary guidance strategy[22], and to design other genes with differing levels of user involvement.

The following figures present examples of how the design tools are used to design a modular mutagenesis segment, and to design an entire gene. Figure 1 is a schematic of the development process for designing a small piece of DNA. *Zuruck* reverse-translates a previously entered amino acid sequence into a general DNA sequence, *map* and *ged* particiapte in the design of the segment. Figure 2 contains two examples of the output from the program *map*, a listing of the restriction sites possible before and after the design process. In this case, the segment to be designed contains the mutation of alanine to serine at position 22 of RNase A (referred to in the figure as 'rm22'), the first in a series of evolutionarily guided mutations. This mutant has been successfully constructed and is presently under study in our laboratories. The manual control option in *ged*, was used to incorporate the new restriction site PaeR7 I , the rest of the codons were chosen to resemble the "wild-type" synthetic gene segment. We chose to insert PaeR7 I because it would be unique to the entire expression plasmid; it was used as a screen in the construction of this particular mutant. In this design, the key element was the

display of possbile restriction sites available to the user through the program *map*; the program *ged* played a minor role.

Figures 3 and 4 depict an actual gene design: that of angiogenin. Angiogenin is a small protein which is approximately 40% homolgous to bovine RNase A, and contains some RNase like characteristics[23]. Figure 3 is an edited log of the session with the user. In the log we see the requests entered by the user to design the sequence and explanatory comments in square brackets. The program first parses the restriction map and DNA information into a table which is a cross reference of the data. Pass 1 is an attempt fo fullfil special reqests which are user defined, in this case there were none. Pass 2 applies the algorithm of choice and attempts the recommended restriction site placements. These two actions could have been done separately through the 'apply' and 'execute' options. Pass 3 then applies known codon usage patterns (in this case from *E. coli*) to further specify the codons. Figure 4 shows a restriction map of the completed design. Using the third algorithm described above, *ged* chooses to incorporate 55 restriction sites from 47 different restriction endonucleases given a possible 76 different restriction endonucleases totalling 352 possible restriction sites.

As presented here, these programs are a useful addition to the assortment of tools available to the biochemist who wishes to design genes. Our plans for the future include a more thorough automatic analysis of the possible combinations of recognition sequences, more options in the display of output, interfaces to other programs that produce alignments and manipulate protein sequences, and modules to assess the problems in synthesizing and constructing the designed gene (e.g. self-complementary regions etc.).

We gratefully acknowledge the Institute for Integrated Systems at the E.T.H. for the use of computer facilities, and the Swiss National Science Foundation.

ABREVIATIONS

The standard genetic representations are used:

A = adenine, G = guanine, C = cytosine, T = thymine,

Y = pyrimidine, either C or T, R = purine, either A or G

M = either A or C, W = either A or T,

S = either C or G, K = either G or T,

D = A, G or T, H = A, C or T,

V = A, C or G, B = C, G or T,

N = any nucleotide base, A, G, C, T.

UNIX is a trademark of A.T. & T.

REFERENCES

1. Ackers, G. K. and Smith, F. R. (1985) *Ann. Rev. Biochem.*, vol. **54**, pp. 597-629.
2. Fersht, A. R., Shi, J-P., Wilkinson, A. J., Blow, D. M., Carter, P., Waye, M. M. Y., and Winter, G. P. (1984) *Angew. Chem.*, vol. **26**, pp. 455-462.
3. Knowles, J. R. (1987) *Science*, vol. **236**, pp. 1252-1258.

4. Gouy, M. and Gautier, C. (1982) *Nucl. Acids Res.*, vol. **10**, pp. 7055-7074.
5. Craik, C. S., Largman, C., Fletcher, T., Roczniak, S., Barr, P. J., Fletterick, R., and Rutter, W. J. (1985) *Science*, vol. **228**, pp. 291-297.
6. Howell, E. E., Villafranca, J. E., Warren, M. S., Oatley, S., and Kraut, J. (1986) *Science*, vol. **231**, pp. 1123-1128.
7. Edge, M. D., Greene, A. R., Heathcliffe, G. R., Meacock, P. A., Schuch, W., Scanlon, D. B., Atkinson, T. C., Newton, C. R., and Markham, A. F. (1981) *Nature*, vol. **292**, pp. 756-762.
8. Fortkamp, E., Rieger, M., Heisterberg-Moutses, G., Schweitzer, S., and Sommer, R. (1986) *DNA*, vol. **5**, pp. 511-517.
9. Itakura, K., Hirose, T., Crea, R., Riggsr, A. D., Heynecker, H. L., Bolivar, F., and Boyer, H. W. (1977) *Science*, vol. **198**, pp. 1056-1063.
10. Crea, R., Kraszewski, A., Hirose, T., and Itakura, K. (1978) *Proc. Nat. Acad. Sci.*, vol. **75**, pp. 5765-5769.
11. Hsing, H. M., Sung, W. L., Broussan, R., Wu, R., and Narang, S. A. (1980) *Nucl. Acids Res.*, vol. **8**, pp. 5753-5756.
12. Suzuki, M., Sumi, S., Hasegawa, A., Nishizawa, T., Miyoshi, K., Wakisawa, S., Miyake, T., and Misoka, F. (1982) *Proc. Nat. Acad. Sci.*, vol. **79**, pp. 2475-2479.
13. Tanaka, S., Oshima, T., Ohsue, K., Ono, T., Mizuno, A., Ueno, A., Nakazato, H., Tsujimoto, M., and Higashi, T. N. (1983) *Nucl. Acids Res.*, vol. **11**, pp. 1707-1723.
14. Edge, M. D., Heathcliffe, G. R., Moore, V. E., Faulkner, N. J., Camble, R., Petter, N. N., Trueman, P. T., Schuch, W., Hennam, J., Atkinson, T. C., Newton, C. R., and Markham, A. F. (1983) *Nucl. Acids, Res.*, vol. **11**, pp. 6419-6435.
15. Ikehara, M., Ohtsuka, E., Tokunaga, T., Taniyama, Y., Iwai, S., Kitano, K., Miyamoto, S., Ohgi, T., Sakuragawa, Y., Fujiyama, K., Ikari, T., Kobayashi, M., Miyake, T., Shibahara, S., Ono, A., Ueda, T., Tanaka, T., Baba, H., Miki, T., Sakurai, A., Oishi, T., Chisaka, O., and Matsubara, K. (1984) *Proc. Nat. Acad. Sci.*, vol. **81**, pp. 5956-5960.
16. Ikehara, M., Ohtsuka, E., Tokunaga, T., Nishikawa, S., Uesugi, S., Tanaka, T., Aoyama, Y., Kikyodani, S., Fujimoto, K., Yanase, K., Fuchimura, K., and Morioka, H. (1986) *Proc. Nat. Acad. Sci.*, vol. **83**, pp. 4695-4699.
17. Nambiar, K. P., Stackhouse, J., Stauffer, D. M., Kennedy, W., Eldredge, J. K., and Benner, S. A. (1984) *Science*, vol. **222**, pp. 1299-1301.
18. Mandecki, W., Mollison, K. W., Bolling, T. J., Powell, B. S., Carter, G. W., and Fox, J. L. (1985) *Proc. Nat. Acad. Sci.*, vol. **82**, pp. 3543-3547.
19. Ferretti, L., Karnik, S. S., Khorana, H. G., Nassal, M., and Oprian, D. D. (1986) *Proc. Nat. Acad. Sci.*, vol. **83**, pp. 599-603.
20. Bodman-von-Beck, S., Schuler, M. A., Jollie, D. R., and Sligar, S. G. (1986) *Proc. Nat. Acad. Sci.*, vol. **83**, pp. 9443-9447.
21. Nambiar, K. P., Stackhouse, J., Presnell, S. R., and Benner, S. A., in *Enzymes as Catalysts in Organic Synthesis*, ed. M. P. Schneider, pp. 325-340, Reidel, 1986.
22. Presnell, S. R., McGeehan, G., Stackhouse, J., and Benner, S. A., *In preparation.*
23. Strydom, D. J., Fett, J. W., Lobb, R. R., Alderman, E. M., Bethune, J. L., Riordan, J. F., and Vallee, B. L. (1985) *Biochemistry*, vol. **24**, pp. 5486-5494.