Object Command Language: a formalism to build molecular models and to analyze structural parameters in macromolecules, with applications to nucleic acids

Jacques Gabarro-Arpa, Jean A.H. Cognet and Marc Le Bret

Equipe de Modélisation Moléculaire, Unité de Biochimie-Enzymologie, Institut Gustave-Roussy (U140 INSERM, URA147 CNRS), France

We have written a programming language OCL (Object Command Language) to solve, in a general way, two recurring problems that arise during the construction of molecular models and during the geometrical characterization of macromolecules: how to move precisely and reproducibly any part of a molecular model in any user-defined local reference axes; and how to calculate standard or user-defined structural parameters that characterize the complex geometries of any macromolecule. OCL endows the user with three main capabilities: the definition of subsets of the macromolecule, called objects in OCL, with a formalism from elementary set theory or lexical analysis; the definition of sequences of elementary geometrical operations, called procedures in OCL, enabling one to build arbitrary three-dimensional (3D) orthonormal reference frames, to be associated with previously defined objects; and the transmission of these definitions to programs that allow one to display, to modify and to analyze interactively the molecular structure, or to programs that perform energy minimizations or molecular dynamics. Several applications to nucleic acids are presented.

Keywords: molecular model building, macromolecular structure parameters, molecular graphics, molecular mechanics, programming language

INTRODUCTION

The idea that subtle variations in the tertiary structure of biological macromolecules can be crucial to their activity,[1,2] has received strong support from the steadily increasing amount of structural knowledge obtained by crystallography and NMR.[1-9] To characterize and analyze these structural details in tertiary structures, many authors[1,10-13] have generalized the usual Cartesian or torsion angle coordinates, and proposed various coordinate systems that describe the positioning of parts of the molecule relative to other parts. In the case of nucleic acids for instance, Dickerson et al.[13] exhaustively reviews definitions and alternative descriptions that are useful for parameterizing in a standard way such structures as double helical DNA molecules or tRNA molecules.

However, as more exotic molecular structures are studied, such as abasic sites,[7] mismatched base pairs[9] or any other unusual DNA structures,[2] more exotic descriptions are needed to analyze, modify or construct these structures. It is difficult to write a program that could incorporate all possible geometrical descriptions since the choice of reference axis systems for the parameterizations is arbitrarily large. Irrespective of their merits no unique parameterization exists that can deal with all types of irregularities arising in complex molecules. Thus we have arrived at the conclusion that the molecular modelists should be offered the capability to resort to their own descriptions if the standard ones appear insufficient.

We have written a programming language, named OCL for *Object Command Language* that provides such an environment to the molecular modelist. The purpose of OCL is to provide the user with three general facilities:

(1) To define any arbitrary subset of a given molecular system as an entity, which is referred to as an *object* in OCL. Objects are built by application of logical operations of elementary set theory, such as union, intersection, and complement, together with simple rules from lexical analysis, upon predefined elementary subsets, such as individual atoms, residues and chains of residues, as they are defined in a Brookhaven

Color plates for this article are on page 162.

Address reprint requests to Dr. Gabarro-Arpa at Equipe de Modélisation Moléculaire, Unité de Biochimie-Enzymologie, Institut Gustave-Roussy (U140 INSERM, URA147 CNRS), 94805 Villejuif Cedex, France.
Received 20 August, 1991; accepted 7 October 1991

Protein Databank file format or in a molecular topology file.[14,15]

(2) To assign an arbitrary 3D orthonormal reference frame to previously defined objects. This is achieved through a sequence of elementary geometrical operations, such as "compute the normal to the plane," "compute the cross product of" or "take the difference of two vectors," applied to the space coordinates of atoms and objects in the structure under consideration. These sequences are referred to as *procedures* in OCL.

(3) To export the definitions of chosen objects and procedures through a unique file towards a molecular graphics program such as MORCAD[16] for visualization, analysis and transformation of the molecule and towards energy refining packages such as MORMIN.[16]

In this paper, we describe OCL as a computer programming language with an original set of instructions to define objects and procedures. We discuss how OCL can serve as support language not only to implement standard definitions for describing macromolecules, such as the recent conventions on DNA or RNA in helical conformation,[13] but also to generate any new definitions that may be required for describing other complex macromolecular conformations. Finally we present several applications of OCL to building molecular models.

OCL REQUIREMENTS

For proper operation, OCL must first read a file that contains a minimal description of the biological macromolecule under study. This description must include the list of names of all atoms, together with the names of the residues to which they belong and the residue number within the molecular chain. Macromolecular systems are often an assembly of discrete parts (hereafter referred to as *molecules* in accordance with Weiner and Kollman[14]); additional information can be the number of the molecule to which a residue belongs. Residues and residue names are the standard amino acids and nucleotides and their usual abbreviations for proteins and nucleic acids, but they can be any part of a drug molecule. Atoms are uniquely referenced by their name, and the residue name to which they belong: O1′ GUA 4, C1 ALA 27, etc.

OCL can read this initial information from two alternative sources: a file in the standard Brookhaven Protein Databank format or a molecular topology file generated with AMBER.[14,15] After entering OCL any of these files can be read with the command:

```
LOAD pdbfile_or_topologyfile
```

(By convention, OCL commands are set in uppercase in this article. Appendix A contains the list of all OCL commands). From this input OCL retains only three types of entities: atoms, residues and molecules; those are the building blocks from which more sophisticated objects can be built.

DEFINITION OF OBJECTS AND VARIABLES WITH OCL

A recurrent difficulty in molecular modeling is to access a particular entity within a large structure. One of the goals of OCL is to facilitate the easy build-up of objects. Any object can be defined interactively with the command:

```
OBJECT object_identifier = expression
```

An object can be declared empty:

```
OBJECT object_identifier = NULL
```

It can be described by lexical analysis with string matching and metacharacters. When OCL is loaded with the topology file of a nucleic acid, by convention,[14] the four-letter code for atom names of the ribose or deoxyribose ends with '. Thus:

```
OBJECT allsugar = "*'" "*'" 0
```

contains all the sugar moieties of all residues in a nucleic acid. The metacharacter * matches any character or string and the number 0 matches any residue number.

```
OBJECT allphosphates = "*" "POM" 0
```

contains all the atoms of all residues called *POM*, i.e. the phosphate residues.[14] More complicated objects can be constructed with operators from set theory: | (union), & (intersection) and ~ (complement) (see Appendix B). Thus:

```
OBJECT allbases = ~ (allsugars | allphosphates)
```

here *allbases* is the set of all atoms in the molecule which are neither part of a sugar nor of a phosphate.

This formalism is sufficient to represent any object. Variables, conditionals and loops are used in OCL to simplify code and repetitions. OCL is a language without formal type declaration statements. Both the value and the type of a variable are attributed with the command SET. The syntax is:

```
SET variable_name = expression
```

The type of a variable is the type of the output of the last executed operation in the expression. OCL supports three basic constant data types: string, floating point and integer; and five types of variables: atom, residue, molecule, string and integer.

```
SET a = 10
SET b = mol 10
SET c = + + first (mol 2)
```

the type of *a* is integer (see Appendix B for a list of the operators available with OCL). The type of *b* is molecule. The type of *c* is residue, since the operator *first* returns the first residue of a molecule.

Objects are indexed in the following way: an object identifier in OCL is composed of a string and an integer number. The string serves as a generic name for a family of objects; members are individually addressed by their identifier. If an identifier bears only a string, by default its number is taken as zero.

```
OBJECT residue 1 = "*" 1
FOR n,2,5
    OBJECT residue n = residue n-1 | "*" n
END
```

in this example with the command FOR (see Appendix A) we have created the family *residue* composed of five objects, which are built of contiguous residues, and each member is embedded in the next one.

The importance of object indexation is that single operations can be applied to a whole family of objects in the recipient programs. In this way, for instance, with a simple transformation applied to a family of objects in succession we can smoothly deform a complex macromolecule.

All expressions in OCL on the righthand side must be a valid combination of parentheses, variables, and operators (see Appendix B), according to the rules of algebra. Otherwise error messages are issued. Definitions of variables or objects are entirely interactive and can be checked at any time with the command: LIST list_of_variable_names.

PROCEDURES: CONSTRUCTION OF 3D REFERENCE FRAMES WITH OCL

A procedure in OCL is a set of rules for constructing a reference frame from an arbitrary list of vectors and objects. Entering a procedure is not interactive, unlike the definition of objects, since a procedure only generates a code for other programs. The construction of 3D reference frames requires OCL to have vectors as the sixth type of variable. Vectors are useful only as procedure input or output and within a procedure. Vectors and atom variables are formally equivalent: both bear a name and are characterized by three cartesian coordinates. Vectors are defined with the OCL command:

```
VECTOR V = X_coordinate Y_coordinate
Z_coordinate
```

or also:

```
VECTOR V = atom1 - atom2
```

The goal of OCL procedures is to cast in a simple formal language the construction of any 3D reference frame within a molecule. Typically, this requires to find a formalism for instructions of the type:

- The *Z*-axis is the unit vector perpendicular to the mean least-squares plane of object *cycle,* and is oriented in the same direction as the vector joining the atoms labeled *up* and *down*.
- The *Y*-axis is along the unit vector obtained as follows: it is the projection in the plane perpendicular to the *Z*-axis of the vector joining the atoms labeled a and b.
- The *X*-axis is the cross product of *Y* and *Z* such that the resulting axes form a righthanded reference frame.
- The origin of the frame is located at the middle point of a and b.

A procedure begins with the command PROCEDURE and finishes with END. It must contain at least 3 lines defining the *X*, *Y*, and *Z*-axes of the reference frame to build, and a line defining the origin. The most simple a procedure has the structure:

```
PROCEDURE name_of_the_procedure (arg1, arg2,
. . . , argn)
  XAX = expression 1
  YAX = expression 2
  ZAX = expression 3
  ORG = expression 4
END
```

XAX, YAX, ZAX and ORG are reserved names for the three unit vectors of the three axes and for the origin of the frame. These names stand for internal vector variables inside a procedure and their values are returned in a coded form at the end of the procedure.

The reference frame described above is constructed with the following OCL procedure:

```
PROCEDURE axbas (cycle,down,up,a,b)
  ZAX = mlsp (cycle)>>(up-down)
  YAX = norm((a-b)->ZAX)
  XAX = YAX^ZAX
  ORG = a%b
END
```

This procedure makes use of the OCL operators mlsp, -, norm, >>, ^ and % (the available operators are listed in Appendix B): mlsp returns a unit vector perpendicular to the mean least-squares plane of the object *cycle;*[17] >> reverses the sign of this vector if its scalar product with (up-down) is negative; -> returns the projection of (b-a) onto the plane perpendicular to ZAX, and norm sets the norm of a vector 1; ^ performs the cross product and % gives the middle point of a and b.

OCL has no declaration statements, and operands must be of a given type (see Appendix B for procedure operators). Thus *cycle* must be an object since it is the operand of mlsp, the other arguments must all be vectors.

ASSIGNMENT OF 3D REFERENCES TO OBJECTS

One of the main goals of OCL is to provide recipient programs with the information needed to determine the position of one object in the reference frame of another. This information may be used either to analyze or to modify the position of one object with respect to the other. To specify this information, objects are attributed a number called a *link* in OCL, which is a pointer to another object:

```
LINK object_id1 [link_number1] , object_id2 [link_
number2]
```

Link numbers are enclosed between brackets, which distinguishes them from object indices. An object's link points not only to another object but also to a reference frame associated with this object. The full command to assign a reference frame to a given object is

```
AXIS object_id [link] = procedure_name (arg1, arg2,
. . . , argn)
```

in this command arguments are transmitted to procedures in the same way as they are transmitted to functions and subroutines in ordinary computer languages, and they can return a value that has been calculated in the procedure.

APPLICATION OF OCL TO IMPLEMENT THE DEFINITION OF STRUCTURE PARAMETERS IN NUCLEIC ACIDS

The parameters needed to describe the geometry of nucleic acids were discussed in a meeting that took place at Churchill College, Cambridge. The overall principle of the conventions that were adopted,[13] is to assign specific reference frames to given objects, i.e., bases or base pairs, and to compute the parameters of the rotations and translations which transform homologous frames one into another.[1,10-13]

All these local reference axes[11,13] can be constructed with OCL. We present below two of these constructions.

Following the specifications given in Figure 2 of Dickerson et al.,[13] the following OCL program along with the procedure axbas previously defined, constructs the local reference frame associated with a base pair:

```
OBJECT cyc =        "N1" a | "C1" a | "N3" a
| "C4" a | "C5" a | "C6" a

OBJECT cyc = cyc | "N1" b | "C1" b | "N3" b
| "C4" b | "C5" b | "C6" b

OBJECT pdb = "*" a | "*" b

SET dw = first ("C5'" a)

SET up  = first ("O3'" a)

SET ca  = first ("C8" a)

SET cb  = first ("C6" b)

AXIS pdb [0] = axbas (cyc,dw,up,ca,cb)
```

In this program a and b are assumed to be integer variables. They correspond to the residue numbers of the pyrimidine and the purine bases respectively, of a given Watson–Crick base pair. Object *cyc* is the first argument entered in the procedure axbas. It is defined in two steps and contains the atoms of the six-membered rings to each base; the *Z*-axis is the unit vector perpendicular to the mean least squares plane of this object. The next arguments *dw* and *up* point to atoms below and above the base pair plane; they define the orientation of the *Z*-axis which is along the 5' to 3' direction of the first strand. The atom variables *ca* and *cb* define the *Y*-direction or long axis which is the projection in the plane perpendicular to *Z* of the line that joins the C6 of the pyrimidine in the first strand to the C8 of the purine in the second strand. They also define the origin of the frame located in the middle point. The *X*-axis is chosen to complete a righthanded reference frame. Color Plate 1a shows the resulting local reference frame as displayed by MORCAD.[16]

Other local reference frames have been proposed as in Figure 1 of Lavery and Sklenar.[11] The following OCL program reproduces a slightly simplified version of this reference frame:

```
PROC axpubas (cyc,a1,a2,a3)
        vax  = mlsp (cyc)
        YAX  = norm ((a1-a2)->vax)
        XAX  = (YAX^vax)>>(a3-a1)
        ZAX  = XAX^YAX
        ORG  = a1-4.469*YAX-0.525*XAX
END

OBJECT base = "*" a

OBJECT cycle = "N1" a | "C1" a | "N3" a | "C4" a
| "C5" a | "C6" a

SET n9 = first ("N9" a)

Set c4  = first ("C4" a)

Set c8  = first ("C8" a)

AXIS base [0] = axpubas (cycle,n9,c4,c8)
```

The vertical axis is perpendicular to the six-membered ring of the purine. The long axis should be parallel to the C6 C8 axis of an ideal base pair.[13,18] Actually the N9 C4 axis is a reasonable approximation; the short axis points towards the major groove in a direction roughly parallel to the C8 N9 axis. Color Plate 1b shows the resulting reference frame as displayed by MORCAD.[16]

MODEL BUILDING OF A NOVEL G.G PAIR WITH OCL

Table 1 lists an OCL run using the procedure axbas described above.

In line 7 the molecular topology file gg, corresponding to an 11-base pair oligonucleotide bearing a G.G mismatch[9] is loaded in OCL. The sequence, residue numbering and base pairing is:

5'd(G2 A4 G6 G8 A10 G12 G14 C16 A18 C20 G22)3'

3'd(C45 T43 C41 C39 T37 G35 C33 G31 T29 G27 C25)5'

Four objects are created: *base 12* and *base 35* hold the mismatched guanines and *cycl 12* and *cycl 35* hold their corresponding six-membered rings, which are used to determine the mean least squares planes.

Lines 14 to 22 define the input variables for the procedures. *lw* and *up* are atoms below and above the base plane and serve to define the orientation of the *Z*-axis; *h1* and *o6* serve to define the direction of the *X*-axis. In lines 18 and 23 the reference frames are attached to objects. In line 25 the two objects containing the bases are linked, and the whole is stored in the file gg.ocl to be transmitted to recipient programs. The QUIT command terminates the run.

The resulting reference frames viewed in Color Plate 2 are calculated for the energy refined structure of Cognet et al.[9] They have been designed for the particular base pairing scheme of the G.G base pair of Color Plate 3e. Color Plate 3 shows how this particular base pairing is built with the OCL protocol.

IMPLEMENTATION OF EXTENDED STRUCTURE PARAMETERS FOR A G.G PAIR

If the base pairing had an ideal geometry, the previously defined homologous reference frames in each base would be roughly parallel. Inside an oligomer distortions from this ideal geometry arise, and by measuring the relative orientation of the homologous reference frames these distortions can be quantified.

Concepts like *propeller twist, opening,* and *buckle,*[13] can

Table 1. OCL run, displayed as it would appear to an operator, each command is preceded by a prompt indicating the line number. Residues 12 and 35 correspond to the mismatched guanines from the 11-mer oligomer of Cognet et al.[9]

```
ocl  1> PROCEDURE axbas (cycle,down,up,a,b)
ocl  2>    ZAX  =  mlsp(cycle)>>(up-down)
ocl  3>    YAX  =  norm((a-b)->ZAX)
ocl  4>    XAX  =  YAX ^ ZAX
ocl  5>    ORG  =  a%b
ocl  6> END
ocl  7> LOAD gg
ocl  8> FOR n m,12,35,23
ocl  9> SET r = res n
ocl 10> OBJECT base n = r
ocl 11> OBJECT cycl n = "N1" n | "C2" n | "N3" n |
"C4" n | "C5" n | "C6" n
ocl 12> END
ocl 13> LIST object base 12,cycl 12

----- base     12 -----
- 1- GUA  12 > O5' C5' H5A' H5B' C4'   H4'   O1' C1'   H1'   N9
- 1-         > C4  N3  C2   N2   HN2A HN2B N1  H1    C6    O6
- 1-         > C5  N7  C8   H8   C3'   H3'   C2' H2A'  H2B'  O3'

----- cycl     12 -----
- 1- GUA  12 > C4  N3  C2   N1   C6    C5

ocl 14> SET dw = first("C5'" 12)
ocl 15> SET up = first("O3'" 12)
ocl 16> SET h1 = first("H1" 12)
ocl 17> SET 06 = first("O6" 12)
ocl 18> AXIS base 12[0] = axbas(cycl 12,dw,up,h1,o6)
ocl 19> SET dw = first("C5'" 35)
ocl 20> SET up = first("O3'" 35)
ocl 21> SET h1 = first("H1" 35)
ocl 22> SET o6 = first("O6" 35)
ocl 23> AXIS base 35[0] = axbas(cycl 35,dw,up,o6,h1)
ocl 24> LINK base 12[0] , base 35[0]
ocl 25> STORE gg.ocl
ocl 26> QUIT
```

easily be generalized with our formalism to mismatches and non-Watson–Crick base pairs. In our example the relative orientation of objects *base 12* and *base 35* (corresponding to the mismatched guanines) in the structure shown in Color Plate 2 can be parametrized as:

Propeller twist	Buckle	Opening
−22.65	11.32	−12.94
Stagger	**Shear**	**Stretch**
0.65	−0.12	−0.14

CONCLUSION

OCL deals and solves two basic problems that arise in molecular modeling: how to define objects with the versatility provided by set theory and lexical analysis, and how to position objects arbitrarily, with respect to each other. We have presented applications to nucleic acids but OCL is a progamming language and is not specific to any macromolecular system. This approach can be used to deal with any molecule: proteins, drugs, polymers, etc.

OCL will grow and become more sophisticated as the range of applications broadens. The first version of OCL contains already a complete set of commands and operations. But a broad range of extensions can be contemplated: dimensioned variables, commands such as WHILE, DO WHILE and FOR as they are in standard C language,[19] and extended geometrical operators could be introduced to facilitate programming in OCL.

OCL has been written in FORTRAN 77 and is implemented on workstations running under Unix, it can be obtained from the authors upon request.

ACKNOWLEDGMENTS

Support of this research through grants from Association pour la Recherche sur le Cancer, Centre National de la Recherche Scientifique, Institut National de la Santé de la Recherche Médicale, Ligue Nationale Contre le Cancer, Ministère de la Recherche et de la Technologie, Université Pierre et Marie Curie are gratefully acknowledged.

APPENDIX A: OCL commands

Command	Arguments	Definition
?		Lists all the command names (on line documentation).
?	command_name	Returns the syntax of the command with an example of its use (on line documentation)
AXIS	object_id [link_number] = procedure_name (list_of_arguments)	Assigns a reference frame to an object
BREAK		Executes the first command after the end of the loop†
CLEAR		Starts OCL anew
CONTINUE		Skips all commands to the end of the loop†
DELETE	list_of_objects	Deleted objects are not stored in external OCL files
ELSE		Else statement†
END		Delimits the end of FOR and IF commands, and procedures
FOR	variable_name , initial_value , final_value {, increment }	Beginning of a loop; the default value for the increment is 1
IF		If statement†
INCLUDE	filename	Reads and executes OCL commands stored in an external file
LINK	object_ida [link_number_a] , object_idb [link_number_b]	Creates a link between two objects
LIST	list_of_items	Lists the value of variables, object contents and compiled procedures
LOAD	filename	Loads a PDB or a molecular topology file
OBJECT	object_id = . . .	Object building
OPTION		Sets the value of various options within OCL
PERMUTAT		Permutes the *x*-, *y*-, *z*-axes in a procedure
PROCEDURE	procedure_name (list_of_arguments)	Beginning of a procedure
QUIT		Exits OCL
READ	filename	Recalls an OCL session previously stored in a file
SET	variable_name = . . .	Sets the value and the type of a variable
STORE	filename	Stores all objects, links and procedures created during an OCL session in an external file, for recipient programs
VECTOR	vector_name = x y z	Sets the value of a vector; vector use is restricted to procedures

†These commands have the same meaning and usage as in C.[19]

APPENDIX B: Description of the operators used in OCL

Symbol	Input	Output
atom	Integer	Pointer to the atom
res	Integer	Pointer to the residue
mol	Integer	Pointer to the molecule
first/last	Object	First/last atom of an object
	Residue	First/last atom of a residue
	Molecule	First/last residue of the molecule

Symbol	Input	Output
name	Atom	Name of an atom
	Residue	Name of a residue
num	Atom	Number of the atom in the system
	Residue	Number of the residue in the system
	Molecule	Number of the molecule in the system

+=	–, integer	The value of the left input incremented by the value of the right operand
-=	–, integer	The value of the left input decremented by the value of the right operand
++	–	Increments the input value by 1
--	–	Decrements the input value by 1
<=	–	Less or equal†
>=	–	Greater or equal†
==	–	Equal†
!=	–	Not equal†
&&	–	And†
||	–	Or†
<	–	Less†
>	–	Greater†
!	–	Not†

The symbol – means any input data type except string

Object operators

Symbol	Input	Output
~	Object	Complement†
&	Two objects	Intersection†
|	Two objects	Union†
'	Two objects	Subtraction a 'b is equivalent to a&(~(a&b))

Procedure operators

Symbol	Input	Output
com	Object	Center of mass of an object
mlsp	Object	Vector perpendicular to the mean least-squares plane[17]
inert1	Object	First inertia axis of an object
inert2	Object	Second inertia axis of an object
inert3	Object	Third inertia axis of an object
norm	Vector	Normalized vector
->	Two vectors	Projection of the first vector onto a plane perpendicular to the second vector
>>	Two vectors	The first vector is oriented such that the scalar product with the second vector is positive
+	Two vectors	Sum of two vectors
-	Two vectors	Difference of two vectors
^	Two vectors	Cross product of two vectors
%	Two vectors	Center of gravity of the input vectors
*	Floating and vector	Product of a vector by a scalar

†These operators have the same meaning and usage as in C.[19]

REFERENCES

1 Fratini, A.V., Kopka, M.L., Drew, H.R. and Dickerson, R.E. Reversible bending and helix geometry in a B-DNA dodecamer: CGCGAATTBrCGCG. *J. Biol. Chem.* 1982, **257**, 14686–14707

2 *Unusual DNA Structures* (R.D. Wells and S.C. Harvey, Eds.) Springer-Verlag, New York, 1988

3 Calladine, C.R. and Drew, H.R. Sequence-specific positioning of core histones on an 860 base-pair DNA. *J. Mol. Biol.* 1987, **195**, 143–173

4 Gartenberg, M.R. and Crothers, D.M. DNA sequence determinants of CAP-induced bending and protein binding affinity. *Nature* 1988, **333**, 824–829

5 Travers, A.A. DNA conformation and protein binding. *Ann. Rev. Biochem.* 1989, **58**, 427–452

6 Fazakerley, G.V., Gabarro-Arpa, J., Le Bret, M., Guy, A. and Guschlbauer, W. The GTm6AC sequence is overwound and bent. *Nucl. Acids Res.* 1989, **17**, 2541–2556

7 Cuniasse, Ph., Sowers, L.C., Eritja, R., Kaplan, B., Goodman, M.F., Cognet, J.A.H., Le Bret, M. Guschlbauer, W. and Fazakerley, G.V. Abasic frameshift in DNA. Solution conformation determined by proton NMR and molecular mechanics calculations. *Biochemistry* 1989, **28**, 2018–2026

8 Kissinger, C.R., Liu, B., Martin-Blanco, E., Kornberg, T.B. and Pabo, C.O. Crystal structure of an engrailed homeodomain-DNA complex at 2.8A resolution: a framework for understanding homeodomain-DNA interactions. *Cell* 1990, **63**, 579–590

9 Cognet, J.A.H., Gabarro-Arpa, J., Le Bret, M., van der Marel, G.A., van Boom, J.H., Guschlbauer, W. and Fazakerley, G.V. The solution conformation of an oligonucleotide containing a G.G mismatch determined by nuclear magnetic resonance and molecular mechanics. *Nucl. Acids Res.* 1991, **24**, 6771–6779

10 von Kitzing, E. and Diekmann, S. Molecular mechanics calculations of dA12.T12 and the curved molecule d(GCTCGAAAAA)4.d(TTTTTCGAGC)4. *Eur. Biophys. J.* 1987, **15**, 13–26

11 Lavery, R. and Sklenar, H. Defining the structure of irregular nucleic acids: conventions and principles. *J. Biomol. Struct. Dynam.* 1989, **6**, 655–667

12 Soumpasis, D.M. and Tung, C.S. A rigorous basepair oriented description of DNA structures. *J. Biomol. Struct. Dynam.* 1988, **6**, 397–420

13 Dickerson, R.E., et al. Definitions and nomenclature of nucleic acid structure parameters. *EMBO J.* 1989, **8**, 1–4

14 Weiner, P.K. and Kollman, P.A. AMBER: Assisted Model Building with Energy Refinement. A general program for modeling molecules and their interactions. *J. of Comp. Chem.* 1981, **2**, 287–303
15 Weiner, S.J., Kollman, P.A., Nguyen, D.T. and Case, D.A. An all atom force field for simulations of proteins and nucleic acids. *J. Comp. Chem.* 1986, **7**, 230–252
16 Le Bret, M., Gabarro-Arpa, J., Gilbert, J.C. and Lemarechal, C. MORCAD and object-oriented molecular modeling package. *J. Chim. Phys.* 1991, **88**, 2489–2496
17 McLachlan, A.D. Least squares fitting of two structures. *J. Mol. Biol.* 1979, **128**, 49–79
18 Arnott, S., Campbell-Smith, P. and Chandrasekaran, R. Atomic coordinates and molecular conformation for DNA DNA, RNA RNA and DNA-RNA helices. *CRC Handbook of Biochem.* 1976, **2**, 411–441
19 Kernighan, B.W. and Ritchie, D.M. *The C Programming Language* Prentice-Hall, Englewood Cliffs, 1978