```
evaluate ( $crystal_lattice.space_group     = "P2(1)2(1)2(1)" )
evaluate ( $crystal_lattice.unit_cell.a     = 61.76           )
evaluate ( $crystal_lattice.unit_cell.b     = 40.73           )
evaluate ( $crystal_lattice.unit_cell.c     = 26.74           )
evaluate ( $crystal_lattice.unit_cell.alpha = 90              )
evaluate ( $crystal_lattice.unit_cell.beta  = 90              )
evaluate ( $crystal_lattice.unit_cell.gamma = 90              )
```

(*a*)

```
define (
   &crystal_lattice.space_group     = P2(1)2(1)2(1) ;
   &crystal_lattice.unit_cell.a     = 61.76 ;
   &crystal_lattice.unit_cell.b     = 40.73 ;
   &crystal_lattice.unit_cell.c     = 26.74 ;
   &crystal_lattice.unit_cell.alpha = 90 ;
   &crystal_lattice.unit_cell.beta  = 90 ;
   &crystal_lattice.unit_cell.gamma = 90 ;
)
```

(*b*)

Fig. 25.2.3.2. Examples of compound symbols and compound parameters. (*a*) The 'evaluate' statement is used to define typed symbols (strings, numbers and logicals). Symbol names are in bold. (*b*) The 'define' statement is used to define untyped parameters. Each parameter entry is terminated by a semicolon. The compound base name 'crystal_lattice' has a number of sub-levels, such as 'space_group' and the 'unit_cell' parameters. 'unit_cell' is itself base to a number of sub-levels, such as 'a' and 'alpha'. Parameter names are in bold.

```
group type = hl object = pa object = pb
   object = pc object = pd end                    (25.2.3.9)
```

where 'pa', 'pb', 'pc' and 'pd' refer to the individual arrays. This group statement indicates to CNS that the specified arrays need to be transformed together when reflection indices are changed, *e.g.* during expansion of the diffraction data to space group *P*1.

25.2.3.3. *Symbols and parameters*

The CNS language supports two types of data elements which may be used to store and retrieve information. *Symbols* are typed variables, such as numbers, character strings of restricted length and logicals. *Parameters* are untyped data elements of arbitrary length that may contain collections of CNS commands, numbers, strings, or symbols.

Symbols are denoted by a dollar sign ($), and parameters by an ampersand (&). Symbols and parameters may contain a single data element, or they may be a *compound* data structure of arbitrary complexity. The hierarchy of these data structures is denoted using a period (.). Figs. 25.2.3.2(*a*) and (*b*) demonstrate how crystal-lattice information can be stored in compound symbols and parameters, respectively. The information stored in symbols or parameters can be retrieved by simply referring to them within a CNS command: the symbol or parameter name is substituted by its content. Symbol substitution of portions of the compound names (*e.g.* '&crystal_lattice.unit_cell.$para') allows one to carry out conditional and iterative operations on data structures, such as matrix multiplication.

25.2.3.4. *Statistical functions*

The CNS language contains a number of statistical operations, such as binwise averages and summations. The resolution bins are defined by a central facility in CNS.

Fig. 25.2.3.3 shows how σ_A, σ_Δ and D (Read, 1986, 1990) are computed from the observed structure factors ('fobs') and the calculated model structure factors ('fcalc') using the CNS statistical operations. The first five operations are performed for the reflections in the test set, while the last three operations expand the results to all reflections. The 'norm' function computes normalized structure-factor amplitudes for the specified arguments. The 'sigacv' function

```
do (eobs=norm(amplitude(fobs)))                              ( test_set )
do (ecalc=norm(amplitude(fcalc)))                            ( test_set )
do (sigmaA=sigacv(eobs,ecalc))                               ( test_set )
do (sigmaD=sqrt(save(amplitude(fobs))^2 (1-sigmaA^2)))       ( test_set )
do (D= sigmaA * sqrt(save(amplitude(fobs))^2  /
          save(amplitude(fcalc)^2)))                         ( test_set )

do (sigmaA=sum(sigmaA*test)/ max(1,sum(test)))               ( all )
do (sigmaD=sum(sigmaD*test)/ max(1,sum(test)))               ( all )
do (D=sum(D*test)/ max(1,sum(test)))                         ( all )
```

Fig. 25.2.3.3. Example for statistical operations provided by the CNS language. 'norm', 'sigacv', 'save' and 'sum' are functions that are computed internally by the CNS program. Binwise operations are in italics ('sigacv', 'save' and 'sum'). The result for a particular bin is stored in all elements belonging to the bin. The σ_A ('sigmaA') parameters are computed in binwise resolution shells. The σ_Δ ('sigmaD') and D parameters are then computed from σ_A and binwise averages involving $|\mathbf{F}_o|^2$ and $|\mathbf{F}_c|^2$. The binwise results are expanded to all reflections by the last three statements. 'test' is an array that is 1 for all reflections in the test set and 0 otherwise. 'sum' is a binwise operation on all reflections with the same partitioning used for the test set.

evaluates σ_A from the normalized structure factors. The 'save' function computes the statistical average

$$\text{save}(f) = \frac{\sum_{hkl} f_{hkl}(w/\varepsilon)}{\sum_{hkl} w}, \qquad (25.2.3.10)$$

where w is 1 and 2 for centric and acentric reflections, respectively, and ε is the statistical weight. The averages are computed binwise, and the result for a particular bin is stored in all selected reflections belonging to the bin.

25.2.3.5. *Symbolic target function*

One of the key innovative features of CNS is the ability to symbolically define target functions and their first derivatives for crystallographic searches and refinement. This allows one conveniently to implement new crystallographic methodologies as they are being developed.

The power of symbolic target functions is illustrated by two examples. In the first example, a target function is defined for simultaneous heavy-atom parameter refinement of three derivatives. The sites for each of the three derivatives can be disjoint or identical, depending on the particular situation. For simplicity, the Blow & Crick (1959) approach is used, although maximum-likelihood targets are also possible (see below). The heavy-atom sites are refined against the target

$$\sum_{hkl} \frac{(|\mathbf{F}_{h_1} + \mathbf{F}_p| - |\mathbf{F}_{ph_1}|)^2}{2v_1} + \frac{(|\mathbf{F}_{h_2} + \mathbf{F}_p| - |\mathbf{F}_{ph_2}|)^2}{2v_2} + \frac{(|\mathbf{F}_{h_3} + \mathbf{F}_p| - |\mathbf{F}_{ph_3}|)^2}{2v_3}. \qquad (25.2.3.11)$$

$\mathbf{F}_{h_1}$, $\mathbf{F}_{h_2}$ and $\mathbf{F}_{h_3}$ are complex structure factors corresponding to the three sets of heavy-atom sites, $\mathbf{F}_p$ represents the structure factors of the native crystal, $|\mathbf{F}_{ph_1}|$, $|\mathbf{F}_{ph_2}|$ and $|\mathbf{F}_{ph_3}|$ are the structure-factor amplitudes of the derivatives, and v_1, v_2 and v_3 are the variances of the three lack-of-closure expressions. The corresponding target expression and its first derivatives with respect to the calculated structure factors are shown in Fig. 25.2.3.4(*a*). The derivatives of the target function with respect to each of the three associated structure-factor arrays are specified with the 'dtarget' expressions. The 'tselection' statement specifies the selected subset of reflections to be used in the target function (*e.g.* excluding outliers), and the 'cvselection' statement specifies a subset of reflections to be used for cross-validation (Brünger, 1992*b*) (*i.e.* the subset is not used

```
associate f_h_1 <atom-selection-1>
associate f_h_2 <atom-selection-2>
associate f_h_3 <atom-selection-3>

target=(
         (abs(f_h_1+f_p)-f_ph_1)^2 / (2*v_1) +
         (abs(f_h_2+f_p)-f_ph_2)^2 / (2*v_2) +
         (abs(f_h_3+f_p)-f_ph_3)^2 / (2*v_3)
       )

dtarget(f_h_1)=
        (
         2*(abs(f_h_1+f_p)-f_ph_1)
           *(f_h_1+f_p)/abs(f_h_1+f_p) / (2*v_1)
         )

dtarget(f_h_2)=
        (
         2*(abs(f_h_2+f_p)-f_ph_2)
           *(f_h_2+f_p)/abs(f_h_2+f_p) / (2*v_2)
         )

dtarget(f_h_3)=
        (
         2*(abs(f_h_3+f_p)-f_ph_3)
           *(f_h_3+f_p)/abs(f_h_3+f_p) / (2*v_3)

tselection=<selection>
cvselection=<selection>
```

(*a*)

```
associate fcalc1 <atom-selection1>
associate fcalc2 <atom-selection2>

target=( abs(fobs) - sqrt(abs(fcalc1)^2+abs(fcalc2)^2))^2 )

dtarget(fcalc1)=( -2 *
            (abs(fobs)-sqrt(abs(fcalc1)^2+abs(fcalc2)^2)) *
              (fcalc1/(sqrt(abs(fcalc1)^2+abs(fcalc2)^2))))

dtarget(fcalc2)=( -2 *
            (abs(fobs)-sqrt(abs(fcalc1)^2+abs(fcalc2)^2)) *
              (fcalc2/(sqrt(abs(fcalc1)^2+abs(fcalc2)^2))))

tselection=<selection>
cvselection=<selection>
```

(*b*)

Fig. 25.2.3.4. Examples of symbolic definition of a refinement target function and its derivatives with respect to the calculated structure-factor arrays. (*a*) Simultaneous refinement of heavy-atom sites of three derivatives. The target function is defined by the 'target' expression. 'f_h_1', 'f_h_2' and 'f_h_3' (in bold) are complex structure factors corresponding to three sets of heavy atoms that are specified using atom selections [equation (25.2.3.7)]. The target function and its derivatives with respect to the three structure-factor arrays are defined symbolically using the structure-factor amplitudes of the native crystal, 'f_p', those of the derivatives, 'f_ph_1', 'f_ph_2', 'f_ph_3', the complex structure factors of the heavy-atom models, 'f_h_1', 'f_h_2', 'f_h_3', and the corresponding lack-of-closure variances, 'v_1', 'v_2' and 'v_3'. The summation over the selected stucture factors ('tselection') is performed implicitly. (*b*) Refinement of two independent models against perfectly twinned data. 'fcalc1' and 'fcalc2' are complex structure factors for the models that are related by a twinning operation (in bold). The target function and its derivatives with respect to the two structure-factor arrays are explicitly defined.

during refinement but only as a monitor for the progress of refinement).

The second example is the refinement of a perfectly twinned crystal with overlapping reflections from two independent crystal lattices. Refinement of the model is carried out against the residual

$$\sum_{hkl} |\mathbf{F}_{\rm obs}| - (|\mathbf{F}_{\rm calc1}|^2 + |\mathbf{F}_{\rm calc2}|^2)^{1/2}. \qquad (25.2.3.12)$$

The symbolic definition of this target is shown in Fig. 25.2.3.4(*b*). The twinning operation itself is imposed as a relationship between the two sets of selected atoms (not shown). This example assumes that the two calculated structure-factor arrays ('fcalc1' and 'fcalc2') that correspond to the two lattices have been appropriately scaled with respect to the observed structure factors, and the twinning fractions have been incorporated into the scale factors. However, a more sophisticated target function could be defined which incorporates scaling.

A major advantage of the symbolic definition of the target function and its derivatives is that any arbitrary function of structure-factor arrays can be used. This means that the scope of possible targets is not limited to least-squares targets. Symbolic definition of numerical integration over unknown variables (such as phase angles) is also possible. Thus, even complicated maximum-likelihood target functions (Bricogne, 1984; Otwinowski, 1991; Pannu & Read, 1996*a*; Pannu *et al.*, 1998) can be defined using the CNS language. This is particularly valuable at the prototype stage. For greater efficiency, the standard maximum-likelihood targets are provided through CNS source code which can be accessed as functions in the CNS language. For example, the maximum-likelihood target function MLF (Pannu & Read, 1996*a*) and its derivative with respect to the calculated structure factors are defined as

```
target = (mlf (fobs, sigma, (fcalc + fbulk),
          d, sigma_delta))
dtarget = (dmlf (fobs, sigma, (fcalc + fbulk),
          d, sigma_delta))                    (25.2.3.13)
```

where 'mlf( )' and 'dmlf( )' refer to internal maximum-likelihood functions, 'fobs' and 'sigma' are the observed structure-factor amplitudes and corresponding σ values, 'fcalc' is the (complex) calculated structure-factor array, 'fbulk' is the structure-factor array for a bulk solvent model, and 'd' and 'sigma_delta' are the cross-validated D and σ_Δ functions (Read, 1990; Kleywegt & Brünger, 1996; Read, 1997) which are precomputed prior to invoking the MLF target function using the test set of reflections. The availability of internal Fortran subroutines for the most computing-intensive target functions and the symbolic definitions involving structure-factor arrays allow for maximal flexibility and efficiency. Other examples of available maximum-likelihood target functions include MLI (intensity-based maximum-likelihood refinement), MLHL [crystallographic model refinement with prior phase information (Pannu *et al.*, 1998)], and maximum-likelihood heavy-atom parameter refinement for multiple isomorphous replacement (Otwinowski, 1991) and MAD phasing (Hendrickson, 1991; Burling *et al.*, 1996). Work is in progress to define target functions that include correlations between different heavy-atom derivatives (Read, 1994).

```
module { compute_unit_cell_volume }
(
   &cell;
   &volume;
)


evaluate ( $cabg.1=cos(&cell.alpha) )
evaluate ( $sabg.1=sin(&cell.alpha) )

evaluate ( $cabg.2=cos(&cell.beta) )
evaluate ( $sabg.2=sin(&cell.beta) )

evaluate ( $cabg.3=cos(&cell.gamma) )
evaluate ( $sabg.3=sin(&cell.gamma) )

evaluate ( &volume=&cell.a * &cell.b * &cell.c *
                   sqrt(1+2*$cabg.1*$cabg.2*$cabg.3
                         -$cabg.1^2-$cabg.2^2-$cabg.3^2) )
```

Fig. 25.2.3.5. Use of compound parameters within a module. This module computes the unit-cell volume (Stout & Jensen, 1989) from the unit-cell geometry. Input and output parameter base names are in bold. Local symbols, such as cabg.1, are defined through 'evaluate' statements. The result is stored in the parameter '&volume' which is passed to the invoking task file or module.