```
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

$$\begin{aligned}\texttt{target} &= (\texttt{mlf (fobs, sigma, (fcalc + fbulk),}\\ &\qquad \texttt{d, sigma\_delta))}\\ \texttt{dtarget} &= (\texttt{dmlf (fobs, sigma, (fcalc + fbulk),}\\ &\qquad \texttt{d, sigma\_delta))} \qquad (25.2.3.13)\end{aligned}$$

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.