required to be in PDB format. An additional input file is the parameter file that governs which plots are to be generated and deals with certain aspects of their appearance.

25.2.6.6. *Output produced*

The output of the program consists of a number of PostScript plots, together with a full listing of the individual parameter values for each residue, with any unusual geometrical properties highlighted. The listing also provides summaries for the protein as a whole. Figs. 25.2.6.2 and 25.2.6.3 show parts of one of the PostScript plots generated, showing the variation of various residue properties along the length of the protein chain. Unusual regions, which are highlighted on these plots, may require further investigation by the crystallographer.

25.2.6.7. *Other validation tools*

PROCHECK is merely one of a number of validation tools that are freely available, some of which are mentioned elsewhere in this volume. The best known are *WHATCHECK* (Hooft *et al.*, 1996), *PROVE* (Pontius *et al.*, 1996), *SQUID* (Oldfield, 1992) and *VERIFY3D* (Eisenberg *et al.*, 1997). Tools such as *OOPS* (Kleywegt & Jones, 1996*b*) or the X-build validation in *QUANTA* (MSI, 1997) provide standard tests on the geometry of a structure and provide lists of residues with unexpected features, which make it easy to check electron-density maps at suspect points.

25.2.7. *MolScript* (P. J. KRAULIS)

25.2.7.1. *Introduction*

Visualization of the atomic coordinate data obtained from a crystallographic study is a necessary step in the analysis and interpretation of the structure. The scientist may use visualization for different purposes, such as obtaining an overview of the structure as a whole, or studying particular spatial relationships in detail. Different levels of graphical abstraction are therefore required. In some cases, the atomic details need to be visualized, while in other cases, high-level structural features must be displayed.

In the study of protein 3D structures in particular, there is an obvious need to visualize structural features at a level higher than atomic. A common graphical 'symbolic language' has evolved to represent schematically hydrogen-bonded repetitive structures (secondary structure) in proteins. Cylinders or helical ribbons are used for α-helices, while arrows or ribbons show strands in β-sheets.

The domain of the *MolScript* program is the production of publication-quality images of molecular structures, in particular protein structures. The implementation of *MolScript* is based on two design principles: First, the program must allow both schematic and detailed graphical representations to be used in the same image. Second, the user must be able to control the precise visual appearance of the various graphics objects in as much detail as possible.

The original version of *MolScript* was written in Fortran77 and produced only PostScript output (Kraulis, 1991). The current version (v2.1.2, as of January 1999) has been completely rewritten in the C programming language. The new version is almost completely compatible with previous versions. The main new features in version 2 are several new output formats, the interactive OpenGL mode (see below) and dynamic memory allocation for all operations.

This section reviews the basic features of *MolScript*. Detailed information about the program, including instructions on how to obtain the software, can be found at the official *MolScript* web site http://www.avatar.se/molscript/, where the online manual is also available.

25.2.7.2. *Input*

The input to *MolScript* consists of the coordinate file(s) in standard PDB format and a script file which describes the orientation of the structures, the graphics objects to display and the graphics state parameters that control the visual appearance of the objects.

The script may be created automatically by the utility program *MolAuto* (see below), or manually by the user in a standard text editor. The script may invoke other external script files or command macros, and it may also contain in-line atomic coordinate data.

25.2.7.3. *Graphics*

The basic model for the execution of *MolScript* is that of a non-interactive image-creating script processor. There are two stages in the execution. First, the script is parsed and the graphics objects are created according to the commands. This stage is essentially independent of the output format. Second, when the end of the script has been reached, the image is rendered from the graphics objects according to the chosen output format.

25.2.7.3.1. *The coordinate system*

The viewpoint in the *MolScript* right-handed coordinate system is always located on the positive z axis, looking towards the origin, with the positive x axis to the right. The user obtains the desired view of the structure by specifying rotations and translations of the atomic coordinates; it is not possible to change the location or the direction of the viewpoint.

There are two main benefits with this scheme: The first is that it is similar to the way we handle objects in everyday life: we do not normally fly around the object, but rather move it about with our hands. The other benefit is that together with the coordinate copy feature it can be used to compose an image containing several geometrically related subunits.

The disadvantage is that the atomic coordinates must be transformed before the creation of the graphics objects. This may complicate the composition of an image where another structure or geometric object is to be included. For example, if two separate structures have been aligned structurally by some external procedure, then the user must take care not to destroy the alignment in the process of setting the viewing transformation.

25.2.7.3.2. *The graphics state*

The graphics state consists of the parameters that determine the exact visual appearance of the graphics objects. The default values of the graphics state parameters are reasonable, so that an image of acceptable quality can be produced quickly. However, to obtain high-quality images which emphasize the relevant structural features, the user must usually fine-tune the rendering by modifying the graphics state parameters appropriately for the various graphics objects.

The graphics state may be modified by using the 'set' command at any point in the script file. The change can have an effect only on graphics commands below that point in the script. When a graphics command is processed, the object is created according to the current values of the graphics state parameters at that exact point in the script file. It is this property of the graphics state that gives the user a very high degree of control over the composition and appearance of the graphics objects.

A new feature in version 2 of *MolScript* is the ability to set the colour of residues on a residue-by-residue basis in schematic secondary-structure representations. It is also possible to set the

references