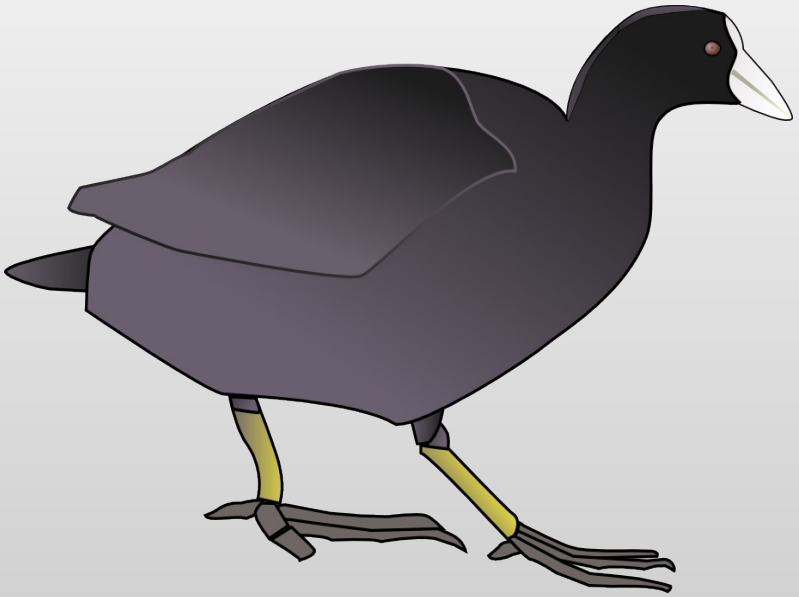
Coot & SHELXL



Paul Emsley

MRC Laboratory of Molecular Biology

Cambridge, UK

August 2014

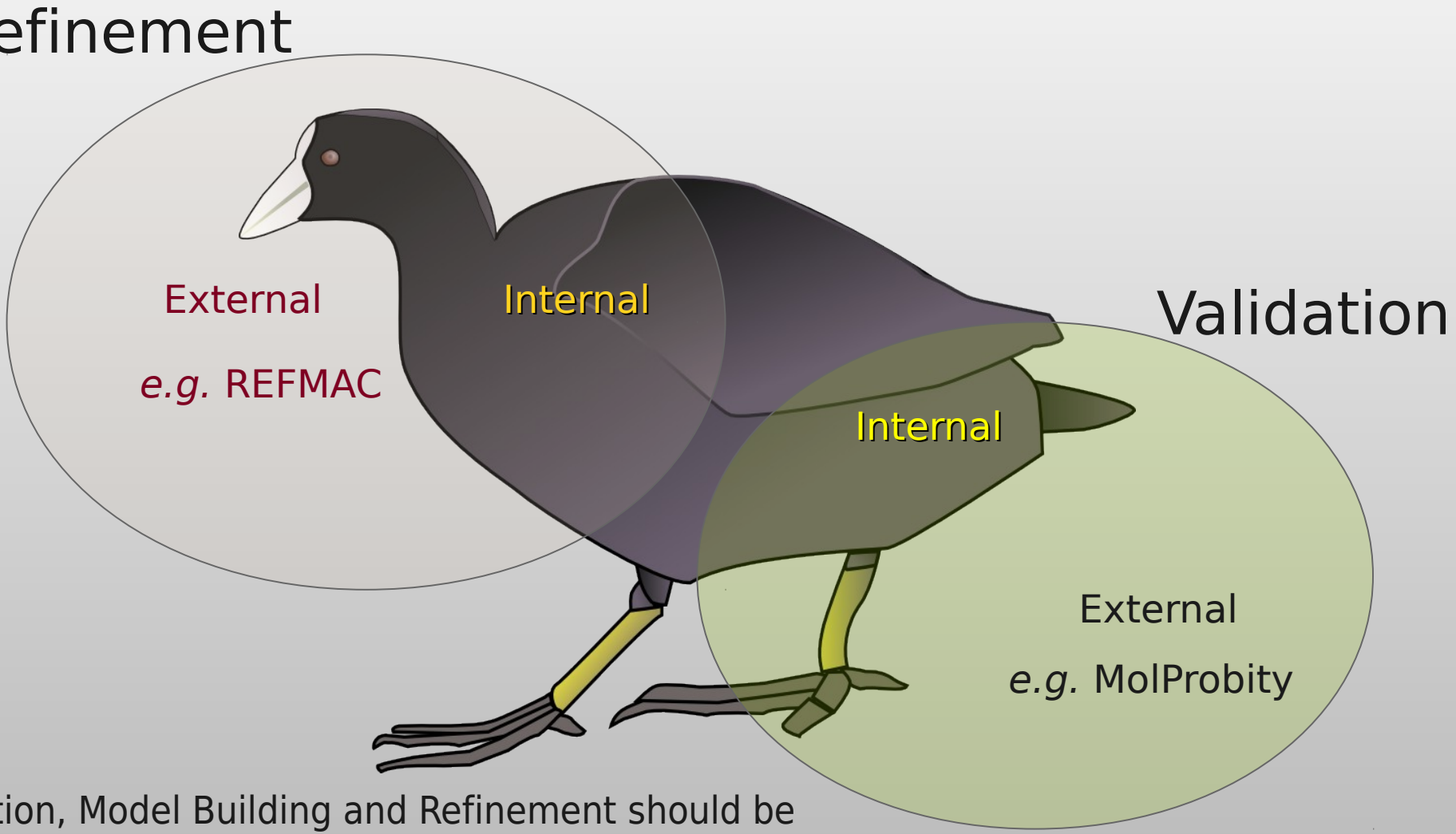
Acknowledgements

- Murshudov Group
  - Andrea Thorn
- George Sheldrick
- Tim Grüne
- Dale Tronrud
- Judit Debreczeni

Coot

- Crystallographic Object-Oriented Tool-kit
- Primarily a tool for the interpretation of electron density generated from X-ray data
  - with tools for modelling:
    - rotate/translate, rotamers,
    - refinement & regularization
    - add, delete
    - ligand fitting
  - A “workhorse”, not a show-pony

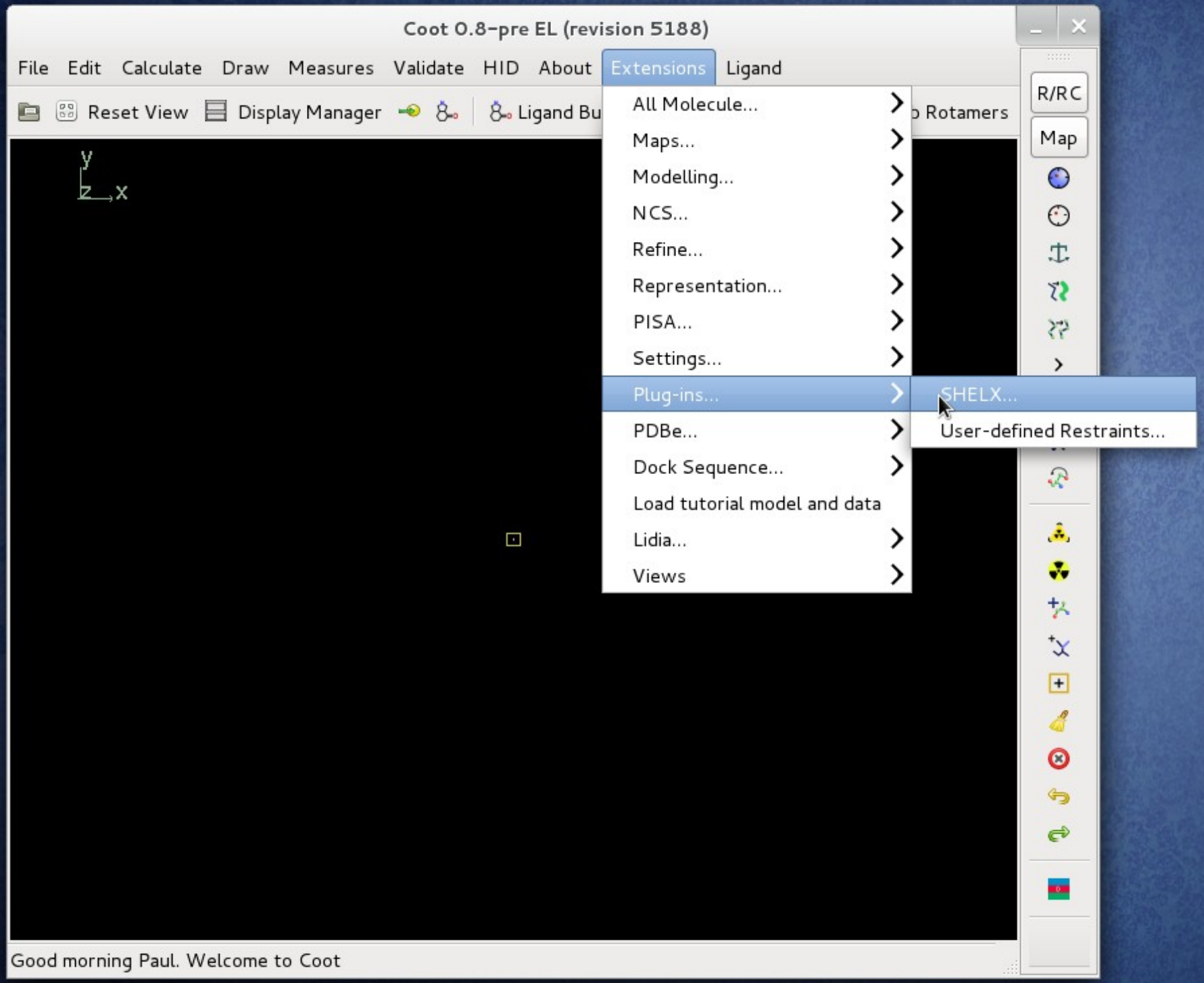
Feature Integration



Validation, Model Building and Refinement should be used together

Coot & SHELXL

- The SHELXL interface is activated from a plug-in
- Using Extensions → Plugins
- or *via* scripting:
  - (activate-plugin-shelx)


Coot 0.8-pre EL (revision 5188)
File
Edit
Calculate
Draw
Measures
Validate
HID
About
Extensions
Ligand
Reset View
Display Manager
Ligand Bu
Rotamers
R/RC
Map
All Molecule...
Maps...
Modelling...
NCS...
Refine...
Representation...
PISA...
Settings...
Plug-ins...
PDBe...
Dock Sequence...
Load tutorial model and data
Lidia...
Views
SHELX...
User-defined Restraints...
Good morning Paul. Welcome to Coot

Coot & SHELXL

- The SHELXL interface is activated from an Extensions plug-in
  - Uses a “project” - the prefix of the file names
  - Project files with the following extensions have special meaning for *Coot*
  - ins: SHELXL instruction file
  - hkl: reflection list (output with HKL4 format)
  - res: as .ins except updated refined values
  - fcf: cif file containing phases
  - lst: listing file, for diagnostics, parsed and interactive
  - pdb: PDB-style coordinates (with WPDB card)

Fitting SHELXL Input into a PDB Container

- The overall idea to to be able to read into *Coot* a .ins file,
  - extracts the molecule
  - displays it
  - optionally manipulate it
  - write out a .ins file

- This turned out to be harder than I thought...

… into a PDB Container

- PDB Container has space group, but no symmetry operators or lattice
  - SHELX is the other way round (of course)

Fitting SHELXL Input into a PDB Container

- We have to *unshelx, i.e.* use a “filter” to convert from SHELX idioms to those of the available storage mechanism
- mmdb container has models which contain chains which contain residues which contain atoms which contain properties such as x,y,z,b_aniso *etc*.
  - which is richer in parts, but less so in others (especially in the parts that are interesting to refinement)
- PARTs become alt-locs
- Large gaps (> 21) in the input residue numbers means new residue set are added to a new chain

… into a PDB Container

- The coordinates are converted to orthogonal coordinates on reading
- The iso-Us of atoms are often based on those of other atoms.
  - There is no natural way to encode this in the PDB container (independent atoms)
  - special atom properties added on-the-fly

Handling Free Variables

- FVAR card read, stored (actually free-variable references) in the occupancy slot and updated as needed on model manipulations via on-the-fly indirection
  - *e.g.* a new alt-conf is created for a side-chain
  - *Coot* recognises that a new FVAR is needed and adds it
    - so that, for example, the occupancies of the new atoms would be 161.0 and -161.0.

Handling AFIX

- AFIX applies constraints for following atoms
  - for which there is no natural storage mechanism in the PDB container
  - each atom is given an attribute “UDD AFIX atom index”, *i.e.* If this is an AFIXed atom, to which “reference” atom does it refer?
- The AFIX values are stored and reproduced on output
  - but not directly editable in *Coot*.

Importing CIF data

- Previously, *Coot* could only read mmCIF data
  - So SHELXL .fcf data files needed to be converted by an external program
- Now *Coot* reads .fcf CIF data directly.

Running a Project

- .ins files are written with time-stamped filenames
- .hkl files are linked
- SHELXL jobs are run in the foreground
  - thus *Coot* stalls while SHELXL is running
    - (easy to program but not desirable from a UI PoV)
- The resulting .lst file is parse for interesting features
  - and creates a GUI for navigation to those sites
- Note: there is no intermolecular bonds...
  - yet...

New Restraints Generation Tools

- The CCP4 programs `cprodrg` and `libcheck` are being retired and/or put into partial-support mode
- CCP4 introduces a new program, ACEDRG
  - based on data derived from the COD
- Coot users can use `pyrogen`
  - based on data derived from CSD (using Mogul)
  - available in Coot 0.8 (to be released Sept 2014)
- Output format is mmCIF
- Convert to SHELX format using
  - `make-shelx-restraints`
  - DFIX, DANG, FLAG, CHIV restraints