Upgrading Fortran source code using automatic refactoring

Dominic Orchard and Andrew Rice

Presented at WRT’13 (co-located with SPLASH 2013)

Context

Programming provides a way to express complex theories as computer models from which predictions are automatically calculated.

However, this approach comes with baggage:

- Programs can be hard to understand, adapt, communicate, verify.
- Validation (is the theory sound?) gets mixed with verification (does the program have bugs?)
- Programs may be specialised to an architecture, data format, solution method, or approximation technique, obscuring the essence of a theory.

Fortran is a popular in science and has had a long history, evolving through many standards.

But many programs have not evolved in-step with Fortran, and contain deprecated features (now) known to impede program verification, maintenance, extension, and understanding.

Approach

Refactoring can help to upgrade code as languages evolve.

We built an automatic refactoring tool, CamFort.

CamFort is open source: www.cl.cam.ac.uk/research/dtg/camfort

Beyond CamFort...

CamFort’s refactorings go towards addressing problem a (improving understanding, maintenance, adaptability, and verification).

The CamFort infrastructure forms the basis of further tools for addressing problem c by including additional type systems and specification languages.

Addressing problem b (without entirely abandoning existing Fortran code) is further work. One approach might be to define a new language that integrates a subset of Fortran with higher-level abstractions, possibly tailored to the scientific domain.