Evolving Fortran types with inferred units-of-measure

Dominic Orchard†, Andrew Rice*, Oleg Oshmyan

† Imperial College London

* University of Cambridge

dorchard.co.uk

ICCS 2015
Mars Climate Orbiter
September 23rd, 1999
Orbital insertion (artist’s impression)

due to a unit mismatch:
foot-pounds (lbf) vs. Newtons (N)

$327.6 million

What actually happened....
(also artist's impression)

NASA/JPL/Corby Waste
Dimensional analysis

(“Great Principle of Similitude”, Isaac Newton, 1686)

$x$ is a length (dimension)

$x$ is in metres (unit of measure)

\[
\text{unit}(x \times y) = (\text{unit } x) \times (\text{unit } y)
\]

\[
\text{unit}(x / y) = (\text{unit } x) / (\text{unit } y)
\]

\[
\text{unit}(x + y) = \text{unit } x = \text{unit } y
\]

\[
\text{unit}(x - y) = \text{unit } x = \text{unit } y
\]

\[
\text{unit}(x^R) = \text{unit}(x)^R
\]
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>562</td>
</tr>
<tr>
<td>Ft. above sea level</td>
<td>2150</td>
</tr>
<tr>
<td>Established</td>
<td>1951</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>4663</strong></td>
</tr>
</tbody>
</table>
Dimensional analysis

= a type system

- House, 1983
  “A proposal for an extended form of type checking of expressions”

- Kennedy, 1994
  “Dimension Types”

- How many (popular) languages have this today?
  F#, Haskell [experimental], C [via extensions]
Fortran - an important target

- Fortran very popular in science
- Evolved considerably over 60 years
- Lots of long-running projects
- Many numerical programs
- A serious need for more verification*

**Automatic tools to the rescue!!!**

*A computational science agenda for programming language research, Orchard, Rice, ICCS 2014*
A recent ISO proposal for Fortran units

unit :: m, s
unit :: mps = m / s
real, unit(m) :: x
real, unit(s) :: t
real, unit(mps):: v
real, unit(mps) :: s
...

v = x / t
s = abs(v)

Follows Fortran tradition of explicit types

• All units must be declared
• All variables must have a unit
• All derived units must have a unique name
‘Explicitness’ tradition hinders evolution

- Two long-running climate modelling projects at Cambridge:
  - (Hybrid 8) 10kloc, 1k variable declarations
  - (Hybrid 4) 8.5kloc, 1.2k variable declarations
Proposal: a lightweight approach

• Type inference
• Implicitly-introduced unit names
• Polymorphism

\[
\text{abs} : \forall u. \text{real}, \text{unit}(u) \rightarrow \text{real}, \text{unit}(u)
\]

• Aid adoption by suggesting annotation points

[Critical variable analysis]

Demo time
CamFort tool...

• Cambridge Fortran research infrastructure [a pre-processor]

• Program analyses, transformations, refactorings
  Upgrading Fortran source code using automatic refactoring, Orchard, Rice, WRT’13

• Type-system extensions (e.g., units-of-measure)

…and project at Cambridge

• (Semi)automated verification and testing

• Integrate with existing working practices

• More sustainable code

A computational science agenda for programming language research, Orchard, Rice, ICCS 2014
Evaluation

- 43 programs from *An Introduction to Computational Physics, Tao Pang, Cambridge University Press, 2006.*

- 50-200 lines in size

- Excluded 6 programs due to MPI or odd syntax
% saving = \left(1 - \frac{\text{size of critical variable set}}{\text{number of variables}}\right) \times 100

Median effort saving = 82.4\% (3sf.)
Evaluation - utility of units

\[
\text{% utility} = \frac{\text{variables assigned a non-unitless unit}}{\text{number of variables}} \times 100
\]

Median utility = 42.8\% (3sf.)
Lessons learned...

• Automatic verification tools are good
  ‣ Inference eases evolution, reduces effort (~ 82% saving)

• Breaking traditions can be good (when they hinder upgrading a code base)

• Units of measure are really worth it!

Download info + tutorial: http://dorchard.co.uk/units

See more: http://dorchard.co.uk/science

Thanks!