Evolving Fortran types with inferred units-of-measure

Dominic Orchard, Andrew Rice, Oleg Oshmyan, Alan Mycroft

PLE’14, Uppsala, Sweden - Monday 28th July
What went wrong?

A unit mismatch!

pounds (\textit{lbf}) instead of Newtons (\textit{N})

cost: $327.6$ million
**Dimensional analysis**

(“Great Principle of Similitude”, Isaac Newton)

\[ x \text{ is a length} \quad (\text{dimension}) \]
\[ x \text{ is in metres} \quad (\text{unit of measure}) \]

\[
\begin{align*}
\text{dim}(x * y) &= \text{dim}(x) * \text{dim}(y) \\
\text{dim}(x / y) &= \text{dim}(x) / \text{dim}(y) \\
\text{dim}(x + y) &= \text{dim}(x) = \text{dim}(y) \\
\text{dim}(x - y) &= \text{dim}(x) = \text{dim}(y) \\
\text{dim}(x^R) &= \text{dim}(x)^R
\end{align*}
\]
NEW CUYAMA

Population 562
Ft. above sea level 2150
Established 1951
TOTAL 4663

photo from Andrew Kennedy’s website http://research.microsoft.com/en-us/um/people/akenn/units/
Dimensional analysis = a type system

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

- Kennedy 1994 (has more of the history)
  - “Dimension Types”

- How many (popular) languages have this today?
  - 1! F#

http://research.microsoft.com/en-us/um/people/akenn/units/
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 can help!
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)
Recent ISO proposal for Fortran units

- All units must be declared
- All variables must have a unit
- All derived units must have a unique name

Follows Fortran type system traditions
‘Explicitness’ tradition hinders evolution

Two long-running climate modelling projects at our University:

● (Hybrid 8) 10kloc - 1k variable declarations
● (Hybrid 4) 8.5kloc - 1.2k variable declarations
Lightweight approach

- Type inference
- Implicitly-introduced unit names
- Polymorphism

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

- Cambridge Fortran research infrastructure
- Program analyses and transformations
- Refactorings
- Type-system extensions
  - Units-of-measure
Live demo
real, unit(m) :: x
real, unit(s) :: t
real :: v
real :: s

\( v = \frac{x}{t} \)

\( s = \text{abs}(v) \)
Initial results on adoption barrier

- Around 10% of variables need explicit annotations

<table>
<thead>
<tr>
<th>Program</th>
<th>Variables</th>
<th>Explicit units required</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>8</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Program 2</td>
<td>20</td>
<td>2</td>
<td>10%</td>
</tr>
<tr>
<td>Program 3</td>
<td>24</td>
<td>2</td>
<td>11%</td>
</tr>
<tr>
<td>Program 4</td>
<td>11</td>
<td>2</td>
<td>5.5%</td>
</tr>
<tr>
<td>Program 5</td>
<td>16</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Program 6</td>
<td>10</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>Program 7</td>
<td>23</td>
<td>3</td>
<td>7.7%</td>
</tr>
</tbody>
</table>
Lessons learned...

- Automatic verification tools are good
- Inference eases evolution, reduces effort
- Breaking traditions can be good (when they hinder upgrading a code base)

Download: https://github.com/dorchard/camfort

See more: http://www.cl.cam.ac.uk/research/dtg/naps/

Thanks!