# *Formal Methods for Industrial Critical Systems at Trinity College, University of Dublin*

(FMICS 2003, Røros, June 7th 2003)

Andrew Butterfield

Trinity College, University of Dublin

`Andrew.Butterfield@cs.tcd.ie`

## Background

Faculty of Engineering

Department of Computer Science

55–60 academic staff, 150 postgradutes, 1500+ undergraduates

6 day degree programmes (4 u/g, 2 p/g), many evening degree+diploma

Foundations and Methods Group

6 staff, 6 post-graduates (11 members in all)
teach about 100 undergraduates

## Foundations and Methods Group

Founded 1992.

Early focus - “Irish School of VDM” ($VDM^{\clubsuit}$)

- some industrial involvement via consultancy firm (K&M Technologies).

Broadening out

- Formal Aspects of CORBA Systems (1997)
- Formalising Handel-C (2000)
- Functional Programming Research
  - Adding OO Concepts to pure lazy languages
  - Formally Modelling I/O Behaviour of pure (lazy) languages (2002)

## *VDM*♣

Not mainstream VDM !

Similar Mathematical Toolkit.

Strong emphasis on explicit postconditions.

Equational reasoning rather than Logic of Partial Functions (LPF).

Strong emphasis on "abstract" algebra concepts as organising principle

> "abstract" means concepts like "monoid", "homomorphism", . . .
>
> but not *too abstract*
>
> — carrier set $A$, functor F, algebra F $A \rightarrow A$, initial algebra, . . .

Akin to a "functional language version" of standard VDM !

## $VDM^\clubsuit$ Specification Example

### Spell Checking Dictionary

$$\begin{array}{rcl}
D \in Dict_0 & = & \mathbb{P}Word \\
\textsf{\textit{inv}}\text{-}Dict_0(D) & \widehat{=} & \forall[isUk]D \\
\textsf{\textit{pre}}\text{-}Ins_0(w)D & \widehat{=} & isUk(w) \\
Ins_0 & : & Word \rightarrow Dict_0 \rightarrow Dict_0 \\
Ins_0(w)D & \widehat{=} & D \cup \{\, w \,\}
\end{array}$$

$$\begin{array}{rcl}
\delta \in Dict_1 & = & Word^\star \\
\textsf{\textit{inv}}\text{-}Dict_1(\delta) & \widehat{=} & \forall[isUk]\delta \\
\textsf{\textit{pre}}\text{-}Ins_1(w)\delta & \widehat{=} & isUk(w) \\
Ins_1 & : & Word \rightarrow Dict_1 \rightarrow Dict_1 \\
Ins_1(w)\delta & \widehat{=} & w : \delta
\end{array}$$

$$\begin{array}{rcl}
\textsf{\textit{retr}}\text{-}Dict_0^1 & : & Dict_1 \rightarrow Dict_0 \\
\textsf{\textit{retr}}\text{-}Dict_0^1\delta & \widehat{=} & \texttt{elems}\ \delta
\end{array}$$

## Proof Obligations

### Invariant Preservation

$$\textit{inv-}Dict_0 D \wedge \textit{pre-}Ins_0(w) D \Rightarrow \textit{inv-}Dict_0(Ins_0(w) D)$$

### (Data) Refinement

$$\textit{inv-}Dict_1 \delta \wedge \textit{pre-}Ins_1(w)\delta \Rightarrow Ins_0(w)(\textit{retr-}Dict_0^1 \delta) = \textit{retr-}Dict_0^1(Ins_1(w)\delta)$$

## Monoids and Homomorphisms

$(M, \star, i)$ is a monoid if:

$$\begin{aligned} \star &: M \times M \to M \\ m \star (n \star p) &= (m \star n) \star p \\ i \star m = m &= m \star i \end{aligned}$$

$h : (M, \star, i) \to (N, *, j)$ is a homomorphism if:

$$\begin{aligned} h(i) &= j \\ h(m_1 \star m_2) &= h(m_1) * h(m_2) \end{aligned}$$

## Example Homomorphisms (and Monoids)

$$\begin{aligned}
\texttt{elems} &: (A^\star, ^\frown, \Lambda) \rightarrow (\mathbb{P}A, \cup, \emptyset) \\
\texttt{len} &: (A^\star, ^\frown, \Lambda) \rightarrow (\mathbb{N}, +, 0) \\
\neg &: (\mathbb{P}A, \cap, A) \rightarrow (\mathbb{P}A, \cup, \emptyset)
\end{aligned}$$

$$\begin{aligned}
\lhd_S &: (A \xrightarrow{p} B, \dagger, \theta) \rightarrow (A \xrightarrow{p} B, \dagger, \theta) \\
\lhd_S &: (\mathbb{P}A, \cup, \emptyset) \rightarrow (\mathbb{P}A, \cup, \emptyset)
\end{aligned}$$

(Note overloading of $\lhd_S$) $\lhd_S$ denotes restriction of argument to contents of set $S : \mathbb{P}A$.

$$\lhd_S(T) = S \cap T$$

In Z, domain restriction is infix, here it is prefix and curried, because this makes the homomorphism evident (Notation matters !)

## Generators and Definitions

Given that a function is a homomorphism on a structure, we can define it completely by simply giving its effects on generator elements.

Many of our monoids have singleton objects as generators.

Defining length and sum this way:

$$\begin{aligned} \mathtt{len} &: (A^\star, ^\frown, \Lambda) \to (\mathbb{N}, +, 0) \\ \mathtt{len}\langle a \rangle &\mathrel{\widehat{=}} 1 \end{aligned}$$

$$\begin{aligned} sum &: (\mathbb{N}^\star, ^\frown, \Lambda) \to (\mathbb{N}, +, 0) \\ sum\langle n \rangle &\mathrel{\widehat{=}} n \end{aligned}$$

This can eliminate a lot of inductive proofs.

## Tool Support for $VDM^{\clubsuit}$

$\LaTeX$

## Tool Support for $VDM^\clubsuit$

LaTeX

Haskell Encoding (animation/execution)

Preliminary integration with QuickCheck (Chalmers, John Hughes)

— will support testing as a means of debugging specifications.

Similar encodings can be done for Clean

Could use the Sparkle Theorem prover for Clean (Nijmegen, Maartens de Mol)

## Recent Work in the "Irish School"

Category Theory

Topos Theory — model of higher order (intuitionistic) logic

Topoi cover:

- Sets and Total Functions (boolean logic)
- Directed Multigraphs (non-boolean)
- Dynamic Systems (endofunctions, non-boolean)
- Dynamic Graphs (sheaves/pre-sheaves, non-boolean)

The latter, and the area of *bigraphs*, is of interest as a foundation for distributed system reasoning techniques.

## Formal Aspects of CORBA Systems (FACS)

CORBA (Common Object Request Broker Architecture)

Object Management Group (OMG) standard for OO middleware

Enterprise Ireland, Basic Research Grant No. SC/97/631

Outcome:

OO-Motivated Process Algebra (OOMPA)

— $\pi$-calculus + class definitions + objects with state

Key ideas

all running agents' code and state associated with a given object
explicit method call and return as part of the calculus syntax
type/sub-typing system to enure correct patterns of usage
a scheme for refining specifications.

Thesis to appear (Autumn '03)

## The Real World Project

Enterprise Ireland, Basic Research Grant No. SC-2002-283

reasoning about the external I/O behaviour of *pure* lazy functional programming languages

look outside the language to the runtime environment of the programs.

Main languages:

> Haskell (`haskell.org`) — uses ADT called a "monad" to handle I/O.
>
> Clean (U. Nijmegen) — uses "unique-types" to handle I/O.

A common approach for both appears feasible

Early case studies have been done.

Goal is to build a hierarchy of models of the I/O runtime of varying levels of detail and complexity, and to provide a method for determining the most suitable for any given application.

## Handel-C Project

Funded by Dean of Research Fund, TCD.

Most “industrially critical” of all research areas.

Hope to get external funding (Celoxica ?)

You have already heard enough about this !, but it is worth noting. . .

— some of the Handel-C semantics has been encoded in Haskell encoding of $VDM^\clubsuit$

— we intend to use this as a QuickCheck case-study.

## Conclusions

Past research largely "foundational".

Emerging trend towards more "applicable" research.

Gradual improvement in research funding.

Irish Govt/Industry showing growing interest in this area

Both Handel-C and Real-World work should lead to FMs for ICSs !