LMF
(Logic and Formal Methods Group)

Luís S. Barbosa

lsb@di.uminho.pt

2005.06.15

Departamento de Informática

Universidade do Minho

The LMF Group

- 20 years tradition in FM (education & research & tools):
  - model-oriented specification
  - functional prototyping (CAMILA)
  - emphasis on calculation (rather than ’invent & verify’)
  - algebraic (structural) *vs* coalgebraic (behavioural) calculi
  - (data-oriented) refinement calculus (SETS)
  - computer security and criptography
  - industrial partnerships

The LMF Group

- but ... if forward software engineering is almost a lost opportunity for FM (with notable exceptions in areas such as safety-critical and dependable computing), its converse still looks a promising area for their application due to the complexity of reversing problems and exponential costs involved
- ... entailed a shift in research focus towards program understanding and reverse engineering

Background:
Industrial projects on data reverse engineering
Strong research on program calculi

Long-term Aim:
Laboratorial infra-structure for software certification

The Approach

- combine standard denotational semantics techniques and model oriented formal design methods(VDM and CAMILA)
- ... with the ‘tradition’ of program calculi driven by (initial or final) type specifications
  - *cf.*, systematic derivation of algorithms in a way that correctness is guaranteed by construction
  - Bird-Meertens ‘school’ [BM87] of mathematics of program construction traced back to Backus FP
  - the Laplace transformation analogue: pointwise - pointfree

Program Understanding

$\boxed{\text{Can the arrows be followed backwards to reconstruct system's specs from legacy code?}}$

$$\mathsf{P} \leadsto [\![P]\!] \dashv S_1 \dashv S_2 ... \dashv S_n$$

- must proceed by inspection (instead of prescription) to build (reasonable aproximations of) $[\![P]\!]$ from non-injective probes
- entails the need for:

Program Understanding

- information gathering (intensive language engineering)
- specific (formal & semi-formal) analysis techniques (*e.g.*, slicing, defusion, type reconstruction, ...)
- code representation, transversal and visualization
- inequational calculi and their reverse application
- again: the calculational transformation to obtain the pointwise denotation of a program, transform it into a subsidiary pointfree denotation, obtain the solution by pointfree algebraic reasoning, and return back to the pointwise level where formal method practitioners are used to express their thought

Program Understanding

Some current research topics

- Software evolution techniques (monadic slicing, slicing by calculation, refactoring)
- Language Engineering (including development of front-ends for relevant legacy code languages)
- Spreadsheets as a Programming Paradigm: Spreadsheets as embedded Domain Specific Languages; Transformation of spreadsheets.
- Combining Strategic Programming and (Higher-Order) Attribute Grammar Programming
- Coinduction by calculation

Program Understanding

Some current research topics

- Constrained datatypes (types as invariants)
- Calculi for reasoning about software architectures
- Algorithm problem solving for pre-university maths curricula