MAP-i: Program Semantics, Verification, and Construction

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI/UM, 2008

Synopsys

Syllabus:

- 12 hours (4 lectures) including assessment:
  - Jan 7th
  - Jan 14th
  - Jan 28th[1]
  - Feb 4th
- Assessment consists of studying and presenting a paper
- Papers should be assigned by Jan 21st
- Presentations will take place on Feb 4th

Handouts:

- several sets of slides
- research papers

[1] No lecture on Jan 21st (away in a research panel in Estonia).

Syllabus in detail

- First lecture (January 7th, 10am-1pm)

  Introduction to the mathematics of program construction. *Correct by verification versus correct by construction. Description versus calculation. Historical perspective on the PF-transform. 'Point-free' notation and reasoning. Rules of the PF-transform.*

- Second lecture (January 14th, 10am-1pm)

  PF-transform: when everything becomes a relation. *Introduction to the binary relation calculus. Taxonomy of binary relations. Functions. Conditions and coreflexives. PF-transform of n-ary relations. Products and Sums. Universal constructions and properties. Galois connections.*

Syllabus in detail

- Third lecture (January 28th, 10am-1pm):
  Alternative a)

  Program construction. *Inductive relations. 'Hiding' relational fixpoints into program combinators. Specs and programs as relational hylomorphisms. Fusion laws. Calculating recursive solutions for hylo-equations.*

  Alternative b)

  Constructive proofs. *A PF-approach to polymorphic type checking. Reynolds' relation on functions. The free-theorem of polymorphism in one equation. Extended static checking (ESC) in the PF-style. Induction-free calculation of preconditions and invariants. Examples.*

Syllabus in detail

- Fourth lecture (February 4th, 10am-1pm)

  First part (two hours): Assessment (paper presentations).
  Second part (one hour): Discussion. Open issues and **hot topics** *in the mathematics of program construction. Research directions in "correct by construction".*

Motivation

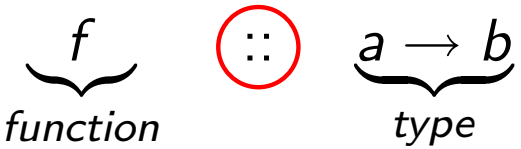
- Much of our **effort** in programming goes into making sure that a number of ("good") **relationships** hold among the **artifacts** we build.

- We have two main ways of **ensuring** that such *good things* happen:
  - **postulate** the relationship + **verify** what has been postulated ("invent & verify")
  - **build** the relationship out of existing valid relationships using an **algebra** of relationships ("correct by construction")

Example — type checking

In functional programming, eg. Haskell:

- Postulate:



- Artifacts: functions (λ-expressions), types (τ-expressions)
- Relationship: *"is of type"*
- Invent & verify: declare $f :: a \longrightarrow b$, define f and wait for the interpreter's reaction
- Correct by construction: start by defining f, then let the interpreter **calculate** its (principal) type; instantiate this if required.

Example — Hoare logic

- Postulate:

$$\{p\}P\{q\}$$

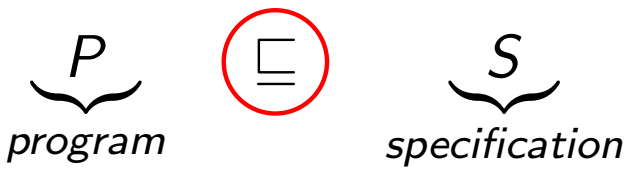
— in fact "the same as"

$$\underbrace{P}_{\textit{program}} \quad :: \quad \underbrace{p \longrightarrow q}_{\textit{predicative type}}$$

- Artifacts: programs (imperative code), pre/post conditions (predicates)
- Relationship: *"pre-condition p ensures post-condition q"*
- Invent & verify: write P, invent p and q and prove that $\{p\}P\{q\}$ holds
- Correct by construction: write P and q; **calculate** the wp for q to hold upon execution of P; obtain p by going stronger, if required.

Example — Refining specifications

- Postulate:



- Artifacts: programs, specifications
- Relationship: *"is a correct implementation of"*
- Invent & verify: given S, invent P and then prove that the semantics of P are more defined than S
- Correct by construction: **calculate** P by transforming S according to some refinement algebra compatible with $\sqsubseteq$.

Example — in discrete maths

- Postulate:

$$\text{function } f \text{ is a bijection}$$

- Artifacts: functions, isomorphisms etc
- Invent & verify: given f, invent its converse f° and then prove the two cancellations

$$\langle \forall\, x \;::\; f^\circ(f(x)) = x \rangle$$
$$\langle \forall\, y \;::\; f(f^\circ(y)) = y \rangle$$

- Correct by construction: from f **calculate** f° (which in general in not a function); both f and f° will be bijective iff a function f° is obtained.

Our target

Aim:

- Our lectures will be devoted to the **calculational**, constructive option illustrated above

However:

- "Traditional" reasoning follows *invent & verify*
- Thinking constructively requires a "turn of mind"

Question:

- Are the logics and calculi we traditionally rely upon up-to-date for such a turn of mind ?