FUNCTORIAL ML

G. Bellè*, C.B. Jay+, E. Moggi*

(*)DISI - Univ. of Genova, Italy, `http://www.disi.unige.it`

(+)SCS - Univ. of Tech. Sydney, Australia, `http://www.socs.uts.edu.au`

- **motivating example**: *map rev*: $L^3X \to l^3X$
  (but potentially more interesting uses)
- **related work**: [BFSS90, RP90], [MFP91, MH95], [CF92], [Jon95, Jeu95], [Jay95a]
- **what is FML**: ML + ...
  - * **functors**, types, schemas, **polymorphism**
  - – terms, type assignment, reduction
- **what are FML functors**:
  - – functors
  - – **shapely functors** [Jay95b]
  - – representations of shapely functors
- **properties of FML** (as expected) [BJM96]:
  - – type inference algorithm
  - – subject reduction, Church-Rosser property
  - – strong normalization (no fix)

A motivating example

- $map\colon \forall F\colon 1.\forall X, Y.(X \to Y) \to FX \to FY$
  - – data polymorphism: F fixed, X, Y vary
  - * shape polymorphism: F varies, X, Y fixed
- $L\colon 1$ the list functor
- $rev\colon \forall X.LX \to LX$ reversing a list

What can be inferred?

- fully explicit

  $map\ L^2\ LX\ LX\ (rev\ X)\colon L^3X \to L^3X$
  $map\ L\ L^2X\ L^2X\ (rev\ LX)\colon L^3X \to L^3X$
- type implicit

  $map\ L^2\ rev\colon L^3X \to L^3X$
  $map\ L\ rev\colon L^3X \to L^3X$

* fully implicit

$map\ rev\colon L^3X \to L^3X$

Approaches

- Charity: F must be explicit
- Haskell type classes: code for map supplied by user
- FML: $F(GX) \neq (F \circ G)(X)$, only isomorphic!

Key idea

- $F(GX) \neq (F \circ G)(X)$, only isomorphic!
- $F(GX)$: F functor, GX data
- $F\langle G\rangle(X)$: $(F \circ G)$ functor, X data
- *bicategory* [Ben67] of functors: composition up to iso (as solutions to domain equations).
- canonical isos as data distribution [Jay95c]

$$
\begin{tikzcd}
F(GX) \arrow[d, dash, "dex"'] & L(L(LX)) \arrow[r, "map\ rev"] \arrow[d, dash, "dex"'] & L(L(LX)) \arrow[d, dash, "dex"'] \\
F\langle G\rangle(X) & L^2(LX) \arrow[r, "map\ rev"'] & L^2(LX)
\end{tikzcd}
$$

$$
H(X) \xrightarrow{map\ f} H(Y)
$$

Functors, types, schema

* functors $\Delta \vdash F\colon m$ of arity m

  $F, G ::= X \mid C \mid \Pi_i^m \mid F\langle\overline{G}\rangle^n \mid \mu^m F$

- types $\Delta \vdash \tau\colon \mathrm{T}$

  $\tau ::= X \mid \tau_1 \to \tau_2 \mid \ldots \mid F(\overline{\tau})$

- schema $\Delta \vdash \sigma$

  $\sigma ::= \tau \mid \forall X.\sigma \mid \forall X\colon m.\sigma$

where $\Delta ::= \_ \mid \Delta, X\colon m \mid \Delta, X\colon \mathrm{T}$ type context.

Functors: arity and "meaning"

- constant functors:
  - $1\colon 0$ unit type
  - $\times, +\colon 2$ binary product and sum

* projection $\Pi_i^m\colon m$, where $i \in m$

  $\Pi_i^m(\overline{X}) \cong X_i$

* composition $F\langle\overline{G}\rangle^n\colon n$, if $F\colon m$ and $G_{i\in m}\colon n$

  $F\langle\overline{G}\rangle^n(\overline{X}) \cong F(G_i(\overline{X})_{i\in m})$

- inductive $\mu^m F\colon m$, if $F\colon m+1$

  $\mu^m F(\overline{X}) \cong \mu Y.F(\overline{X}, Y)$

What are FML functors?

* a refinement of type-constructors:
  with *some* additional structure/properties

  e.g. functors, shapely functors, ...

- **not** ML functors:
  but could be *encoded* as ML structures

  F^*: `sig_m` when F: m, where

```
sig_1 = sig
  type 'X C ; (* or eqtype *)
  val M : ('X->'Y)->'X C->'Y C ;
  ... end;
```

  but **no** inverse *decoding*.

- **not** Haskell/Gofer type classes:
  but could be *encoded* as elements of a type class (same idea as above).

Slogan: FML functors form an **abstract** datatype of type constructors!

Some semantics for $F\colon 1$ in Set

- $F\colon \mathbf{Set} \to \mathbf{Set}$ functor

  composition and projections up to equality
  no implicit shape polymorphism (nor $\mu^m F$)

- $F\colon \mathbf{Set} \to \mathbf{Set}$ functor with relational action

$$\begin{tikzcd} X_1 \arrow[r, "f"] \arrow[d, dash, "R"'] & Y_1 \arrow[d, dash, "S"] \\ X_2 \arrow[r, "g"'] & Y_2 \end{tikzcd} \quad \text{implies} \quad \begin{tikzcd} FX_1 \arrow[r, "Ff"] \arrow[d, dash, "FR"'] & FY_1 \arrow[d, dash, "FS"] \\ FX_2 \arrow[r, "Fg"'] & FY_2 \end{tikzcd}$$

- $F\colon \mathbf{Set} \to \mathbf{Set}$ shapely functor, i.e.

$$\begin{tikzcd} F(X) \arrow[r, "data_X"] \arrow[d, "F(!)"'] \arrow[dr, phantom, "\lrcorner"] & L(X) \arrow[d, "L(!)"] \\ S \cong F(1) \arrow[r, "\# = data_1"'] & L(1) \cong N \end{tikzcd}$$

  $F(X) \cong \Sigma s\colon S.X^{\#s}$ naturally in X

* F repr. $\langle S \in \mathbf{Set}, \#\colon S \to N \rangle$ of shapely functor

$$F(X) == \Sigma s\colon S.X^{\#s} \qquad \Pi_0^1 == \langle 1, \lambda\_\colon 1.1 \rangle$$

  composition and projections only up to iso
  implicit shape polymorphism (and $\mu^m F$)

Terms a la Church

- terms $\Delta; \Gamma \vdash t\colon \sigma$

  $t\colon := x \mid c \mid \lambda x.t \mid t_1\ t_2 \mid \text{let } x = t_1 \text{ in } t_2$

  where $\Gamma\colon := \_ \mid \Gamma, x\colon \sigma$ (if $\Delta \vdash \sigma$) term context.

Rules for type assignment

$$(c)\ \frac{\Delta; \Gamma \vdash}{\Delta; \Gamma \vdash c\colon \sigma_c} \qquad (x)\ \frac{\Delta; \Gamma \vdash}{\Delta; \Gamma \vdash x\colon \sigma}\ \sigma = \Gamma(x)$$

$$(^*\mathrm{App}_n)\ \frac{\Delta; \Gamma \vdash t\colon \forall X\colon n.\sigma \quad \Delta \vdash F\colon n}{\Delta; \Gamma \vdash t\colon \sigma\{F/X\}}$$

$$(^*\Lambda_n)\ \frac{\Delta, Y\colon n; \Gamma \vdash t\colon \sigma\{Y/X\}}{\Delta; \Gamma \vdash t\colon \forall X\colon n.\sigma}\ Y \notin \mathrm{DV}(\Delta)$$

$$(\mathrm{App})\ \frac{\Delta; \Gamma \vdash t\colon \forall X.\sigma \quad \Gamma \vdash \tau}{\Delta; \Gamma \vdash t\colon \sigma\{\tau/X\}}$$

$$(\Lambda)\ \frac{\Delta, Y\colon \mathrm{T}; \Gamma \vdash t\colon \sigma\{Y/X\}}{\Delta; \Gamma \vdash t\colon \forall X.\sigma}\ Y \notin \mathrm{DV}(\Delta)$$

$$(\mathrm{app})\ \frac{\Delta; \Gamma \vdash t\colon \tau_1 \to \tau_2 \quad \Delta; \Gamma \vdash t_1\colon \tau_1}{\Delta; \Gamma \vdash (t\ t_1)\colon \tau_2}$$

$$(\lambda)\ \frac{\Delta; \Gamma, y\colon \tau_1 \vdash t\{y/x\}\colon \tau_2}{\Delta; \Gamma \vdash (\lambda x.t)\colon \tau_1 \to \tau_2}\ y \notin \mathrm{DV}(\Gamma)$$

$$(\mathrm{let})\ \frac{\Delta; \Gamma \vdash t_1\colon \sigma_1 \quad \Delta; \Gamma, y\colon \sigma_1 \vdash t_2\{y/x\}\colon \sigma_2}{\Delta; \Gamma \vdash \text{let } x = t_1 \text{ in } t_2\colon \sigma_2}\ y \notin \mathrm{DV}(\Gamma)$$

Constants and their schema

$$\mathrm{map}_m \ : \ \forall F\colon m.\forall \overline{X}, \overline{Y}.(X_i \to Y_i)_{i\in m} \to F(\overline{X}) \to F(\overline{Y})$$

$$\mathrm{pex}_{m,i} \ : \ \forall \overline{X}.X_i \to \Pi_i^m(\overline{X})$$
$$\mathrm{pin}_{m,i} \ : \ \forall \overline{X}.\Pi_i^m(\overline{X}) \to X_i$$

$$\mathrm{dex}_{m,n} \ : \ \forall F\colon m.\forall \overline{G}\colon n.\forall \overline{X}.F(G_i(\overline{X})_{i\in m}) \to F\langle \overline{G}\rangle^n(\overline{X})$$
$$\mathrm{din}_{m,n} \ : \ \forall F\colon m.\forall \overline{G}\colon n.\forall \overline{X}.F\langle \overline{G}\rangle^n(\overline{X}) \to F(G_i(\overline{X})_{i\in m})$$

$$\mathrm{intro}_m \ : \ \forall F\colon m+1.\forall \overline{X}.F(\overline{X}, \mu^m F(\overline{X})) \to \mu^m F(\overline{X})$$
$$\mathrm{fold}_m \ : \ \forall F\colon m+1.\forall \overline{X}, Y.(F(\overline{X}, Y) \to Y) \to \mu^m F(\overline{X}) \to Y$$

$$\mathrm{un} \ : \ 1$$
$$\mathrm{pair} \ : \ \forall X_0, X_1.X_0 \to X_1 \to X_0 \times X_1$$
$$\mathrm{pi}_j \ : \ \forall X_0, X_1.X_0 \times X_1 \to X_j$$

$$\mathrm{in}_j \ : \ \forall X_0, X_1.X_j \to X_0 + X_1$$
$$\mathrm{case} \ : \ \forall X_0, X_1, Y.(X_0 \to Y) \to (X_1 \to Y) \to X_0 + X_1 \to Y$$

$$\mathrm{fix} \ : \ \forall X.(X \to X) \to X$$

- **main result**: ML-like type inference algorithm.

Type-free Reduction

$$
\begin{array}{rcl}
(\lambda x.t_2)\ t_1 & > & t_2\{t_1/x\} \\
\text{let } x = t_1 \text{ in } t_2 & > & t_2\{t_1/x\}
\end{array}
$$

$$
\begin{array}{rcl}
\text{pi}_j\ (\text{pair } t_0\ t_1) & > & t_j \\
\text{case } f_0\ f_1\ (\text{in}_j\ t) & > & f_j\ t
\end{array}
$$

$$
\begin{array}{rcl}
\text{pin}_{m,i}\ (\text{pex}_{m,i}\ t) & > & t \\
\text{din}_{m,n}\ (\text{dex}_{m,n}\ t) & > & t
\end{array}
$$

$$
\begin{array}{rcl}
\text{map}_m\ f_{k\in m}\ (\text{pex}_{m,i}\ t) & > & \text{pex}_{m,i}\ (f_i\ t) \\
\text{map}_n\ f_{k\in n}\ (\text{dex}_{m,n}\ t) & > & \text{dex}_{m,n}\ (\text{map}_m\ (\text{map}_n\ \overline{f})_{i\in m}\ t) \\
\text{map}_m\ f_{i\in m}\ (\text{intro}_m\ t) & > & \text{intro}_m\ (\text{map}_{m+1}\ \overline{f}\ (\text{map}_m\ \overline{f})\ t) \\
\text{map}_2\ f_0\ f_1\ (\text{pair } t_0\ t_1) & > & \text{pair}\ (f_0\ t_0)\ (f_1\ t_1) \\
\text{map}_2\ f_0\ f_1\ (\text{in}_j\ t) & > & \text{in}_j\ (f_j\ t) \\
\text{map}_0\ \text{un} & > & \text{un}
\end{array}
$$

$$
\text{fold}_m\ f\ (\text{intro}_m\ t)\ >\ f\ (\text{map}_{m+1}\ (\lambda x.x)_{i\in m}\ (\text{fold}_m\ f)\ t)
$$

main results

- Church-Rosser property;
- Subject reduction;
- Strong normalization (when no fix).

Sample reductions

- $\mathrm{fold}_0\ f\ (\mathrm{intro}_0\ t) > f\ (\mathrm{map}_1\ (\mathrm{fold}_0\ f)\ t)$

$$\begin{tikzcd}
F(\mu^0 F) \arrow[r, "\mathrm{intro}_0"] \arrow[d, "\mathrm{map}_1(\mathrm{fold}_0 f)"'] & \mu^0 F \arrow[d, "\mathrm{fold}_0 f"] \\
F(Y) \arrow[r, "f"'] & Y
\end{tikzcd}$$

- $\mathrm{map}_1\ f\ (\mathrm{intro}_1\ t) > \mathrm{intro}_1\ (\mathrm{map}_2\ f\ (\mathrm{map}_1\ f)\ t)$

$$\begin{tikzcd}
F(X, \mu^1 F(X)) \arrow[r, "\mathrm{intro}_1"] \arrow[d, "\mathrm{map}_2(f, \mathrm{map}_1 f)"'] & \mu^1 F(X) \arrow[d, "\mathrm{map}_1 f"] \\
F(Y, \mu^1 F(Y)) \arrow[r, "\mathrm{intro}_1"'] & \mu^1 F(Y)
\end{tikzcd}$$

Conclusions

- FML functors as ADT
  - several *implementations* possible
  - a smooth way to extend ML?
    NESL, core-ML $\subset$ FML
  - but rather low-level, e.g.
    lists $L = \mu^1 F$ where $F = +\langle 1\langle\rangle^2, \times\rangle$
    nil $= (\mathrm{intro}_1 \circ \mathrm{dex}_{2,2} \circ \mathrm{in}_0 \circ \mathrm{dex}_{0,2})$ un: LX
- study FML terms a la Curry: richer models, better optimizations [HM95]
- canonical isos inference problem
- FML as intermediate language for analysis of access to data (canonical isomorphisms) [PJ91, Ler92]

References

[Ben67] J. Benabou. *Introduction to bicategories*, volume 47. Springer, 1967.

[BFSS90] E.S. Bainbridge, P.J. Freyd, A. Scedrov, and P.J. Scott. Functorial polymorphism. *Theoretical Computer Science*, 70:35–64, 1990.

[BJM96] G. Bellè, C. B. Jay, and E. Moggi. Functorial ML. available from ftp://ftp.disi.unige.it/person/MoggiE/functorial_ml.dvi, 1996.

[CF92] J.R.B. Cockett and T. Fukushima. About **charity**. Technical Report 92/480/18, University of Calgary, 1992.

[HM95] R. Harper and G. Morrisett. Compiling polymorphism using intensional type analysis. In *Conference Record of POPL '95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 130–141, San Francisco, California, January 1995.

[Jay95a] C.B. Jay. Polynomial polymorphism. In R. Kotagiri, editor, *Proceedings of the Eighteenth Australasian Computer Science Conference: Glenelg, South Australia 1–3 February, 1995*, volume 17, pages 237–243. A.C.S. Communications, 1995.

[Jay95b] C.B. Jay. A semantics for shape. *Science of Computer Programming*, 25:251–283, 1995.

[Jay95c] C.B. Jay. Shape analysis for parallel computing. In *Parallel Computing Workshop '95 at Fujitsu Parallel Computing Centre, Imperial College*, 1995.

[Jeu95] J. Jeuring. Polytypic pattern matching. In *Conference on Functional Programming Languages and Computer Architecture*, pages 238–248, 1995.

[Jon95] M.P. Jones. A system of constructor classes: overloading and implicit higher-order polymorphism. *J. of Functional Programming*, 5(1), 1995.

[Ler92] X. Leroy. Unboxed objects and polymorphic typing. In *19th Symp. on Principle of Programming Languages*. ACM Press, 1992.

[MFP91] E. Meijer, M. Fokkinga, and R. Paterson. Functional programming with bananas, lenses, envelopes and barbed wire. In J. Hughes, editor, *Procceding of the 5th ACM Conference on Functional Programming and Compter Architecture*, volume 523 of *LNCS*, pages 124–44. Springer Verlag, 1991.

[MH95] E. Meijer and G. Hutton. Bananas in space: extending fold and unfold to exponential types. In *Procedings 7th International Conference on Functional Programming and Computer Architecture, San Diego, California, June 1995*. ACM Press, 1995.

[PJ91] S. Peyton Jones. Unboxed values as first-class citizens. In *Functional Programming and Computer Architecture*, volume 523 of *LNCS*, 1991.

[RP90] J. Reynolds and G.D. Plotkin. On functors expressible in polymorphic lambda-calculus. In G. Huet, editor, *Logical Foundations of Functional Programming*. Addison-Wesley, 1990.