Meanings of Computation*

Michael Stone — October 26, 2006

mstone@sccs.swarthmore.edu — **Swarthmore College**

1 What *is* a Program?

Programs are supposed to be like Legos: small and simple pieces *composed* into large, complex towers.

Program as Legos:

Categories are like mathematical boxes of Legos: they have lots of pieces called *arrows* (*1-cells*)and every arrow $f : A \to B$ has a source $sf = A$ and a target $tf = B$ (which are special arrows called *identity* arrows (*0-cells*). Just like with Legos, two compatible arrows f, g can be composed to form the new arrow $f;g$ when $tf = sg$. This composition is associative and obeys several identity laws.

If programs are like Legos, what arrow should a given piece of source code denote?

2 The Problem of Side-Effects

Programs have side-effects; (mathematical) functions have *only* input and output. Therefore, programs which compute the same function may be unequal:

Same Functions, Different Programs

```
input integer x          input integer x
if (x is 1)              if (x is not 1)
   launch nukes             cure cancer
output (x + 1)           output (x + 1)
```

One solution [3] to is to separate functions (which are arrows of type $A \to B$) from computations with effect C by assigning computations to the type $(A \to CB)$.

For example, the function $f(x) = x + 1$ on the integers would have type $(\mathbb{Z} \to \mathbb{Z})$, the computation which lauches missiles would now have type $(\mathbb{Z} \to \texttt{Bomb}\,\mathbb{Z})$, and the computation curing cancer would have type $(\mathbb{Z} \to \texttt{Cure}\,\mathbb{Z})$.

3 Representing Side-Effects

Now that programs with effect C are arrows $(A \to CB)$ in a category P, we need a process for lifting functions into effect free computations and some categorical glue to make composition well-typed.

The lifting is easy: for each 0-cell T in P define an arrow (1-cell) $\ell_T : T \to CT$. This family of arrows is written as a single *natural transformation* $\ell : \mathbb{1}_P \to C$ which takes the identity *functor* $\mathbb{1}_P$ to the functor C.

Now for composition: we need categorical glue, denoted by a superscript *, so that if $e_2 : Y \to CZ$ is a program then $e_2^* : CY \to CZ$ is its *adaptation* to a program suitable for composition, as in:

Composition of Programs

No side effects	$X \xrightarrow{p} Y \xrightarrow{q} Z$
Side effects	$X \xrightarrow{p} CY \quad ??? \quad Y \xrightarrow{q} CZ$
Adapted with glue	$X \xrightarrow{p} CY \xrightarrow{q^*} CZ$
Defining the glue	$X \xrightarrow{p} CY \xrightarrow{Cq} CCZ \xrightarrow{\mu} CZ$

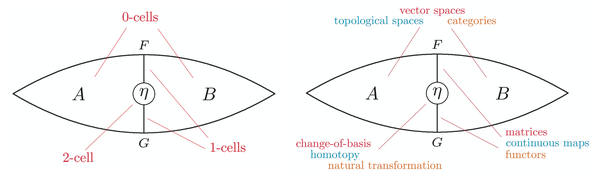
Thus we define the glue * with a more primitive natural transformation $\mu : C^2 \to C$ and the existing computation-forming functor C.

4 2-Categories

As the previous diagram indicates, the glue * is defined in terms of a new natural transformation μ from the functor C^2 to C. Together, C, ℓ, and μ form a structure called a *monad*. Monads, change-of-monad, and change-of-category transformations form the *0-cells*, *1-cells*, and *2-cells*, respectively, of a structure called a *2-category* [2], the elements of which can be depicted by *pasting diagrams* [1].

The diagram below pictures several interpretations of a single 2-cell f mapping the composition of a single 1-cell $F : A \to B$ into the composition containing a single 1-cell $G : A \to B$ in the contexts of linear algebra, topology, and category theory:

Picture of a 2-Cell and Example Interpretations

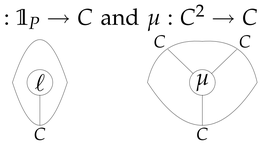


As the name "pasting diagram" suggests, elements in 2-categories can be composed in two directions; if categories are like boxes of Legos, then 2-categories are like jigsaw puzzles. More formally, a 2-category is a collection of 2-cells (this collection includes the 1-cells and 0-cells as "identity" elements) together with composition rules $(a;\ b, \frac{a}{b})$, source functions s_1, s_2, and target functions t_1, t_2. Like with categories, both composition rules are required to obey certain associativity, identity, and interchange laws.

5 Monads

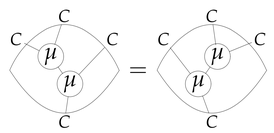
A monad in a category P is given by three components: a functor $C : P \to P$ and a pair of natural transformations

$$\ell : \mathbb{1}_P \to C \text{ and } \mu : C^2 \to C$$



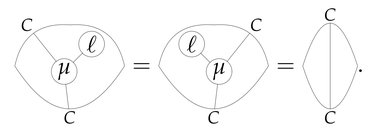
satisfying the "associativity" identity:

$$\frac{\mu;\, C}{\mu} \quad = \quad \frac{C;\, \mu}{\mu}$$



and the "neutral-element" identity:

$$\frac{C;\, \ell}{\mu} \quad = \quad \frac{\ell;\, C}{\mu} \quad = \quad C$$



Intuitively, the 2-cells of the monad are the essential components of our ability to sequence or compose programs; however, some examples will illustrate the flavor of the technique:

The simplest monad is the trivial monad in which each of C, ℓ, and μ is an identity functor (natural transformation). This monad's semantics disallow all side-effects. Other examples include the powerset monad (where computations non-deterministically return sets of results) and the probability monad (where computations return probability distributions over the result space). See Wadler [4] for more detailed (computer-science flavored) examples.

6 Future Work

While some attention has been given in the literature to the interaction of multiple monadic notions of computation, little attention has been devoted (in the computer science community) to the underlying 2-categorical structure of the monads and their change-of-monad and change-of-category transformations. This rich structure is well worth further exploration by a computationally-oriented mind.

References

[1] André Joyal and Ross Street. The geometry of tensor calculus, i. *Advances in Mathematics*, 88:55–112, 1991.

[2] Saunders Mac Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 1998.

[3] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1), 1991.

[4] P. L. Wadler. Comprehending monads. In *Proceedings of the 1990 ACM Conference on LISP and Functional Programming, Nice*, pages 61–78, New York, NY, 1990. ACM.

The complete bibliography for this project is available at:

`http://www.citeulike.org/user/mstone/`

For more information, please see

`http://www.sccs.swarthmore.edu/~mstone/summer2006/`

* This research was supported by a grant from the Howard Hughes Medical Institute.

Special thanks are also due to Prof. Thomas Hunter for supervising the construction of this poster and my related summer work.