Relating Idioms, Arrows and Monads from Monoidal Adjunctions @ SYCO I

Exequiel Rivas
September 2018

Team $\pi r^2$, Inria
The basic idea behind the semantics of programs described below is that a program denotes a morphism from $A$ to $TB$.

E. Moggi 1989
Semantics of effectful programming languages

The basic idea behind the semantics of programs described below is that a program denotes a morphism from $A$ to $TB$.

E. Moggi 1989

Moggi used monads for an unified treatment of effects.

\[ \text{Id} \xrightarrow{\eta} T \leftarrow \mu T \circ T \]

His usages follows:

- $\eta$ lifts values to effectful computations, i.e. `return`.
- $\mu$ composes two effects sequentially, i.e. `;`. 
Monads can be *internalised* as an *interface*.

```haskell
class Functor m ⇒ Monad m where

    return :: a → m a

    (>>=) :: m a → (a → m b) → m b
```

The state monad *State* comes with operations

```haskell
get :: State Int , put :: Int → State ()
```

Computations written using these operations and the interface.

```haskell
get >>= λi → if i ≡ 0 then return False
    else put 1 >>= \_ → return True
```
Monads (as interfaces) has been generalised...
Arrows and applicative functors

Monads (as interfaces) has been generalised...

Providing more control over the computations.

```haskell
class Functor f ⇒ Idiom f where
    pure :: a → f a
    (⊛) :: f (a → b) → f a → f b
```
Arrows and applicative functors

Monads (as interfaces) has been generalised...

Providing more control over the computations.

\[
\text{class } \textit{Functor } f \Rightarrow \textit{Idiom } f \textit{ where }
\]
\[
\text{pure} :: a \rightarrow f \ a
\]
\[
(\triangleright) :: f (a \rightarrow b) \rightarrow f \ a \rightarrow f \ b
\]

\[
\text{class } \textit{Arrow } (\rightleftharpoons) \textit{ where }
\]
\[
\text{arr} :: (x \rightarrow y) \rightarrow x \rightleftharpoons y
\]
\[
(\ggg) :: (x \rightleftharpoons y) \rightarrow (y \rightleftharpoons z) \rightarrow x \rightleftharpoons z
\]
\[
\text{first} :: (x \rightleftharpoons y) \rightarrow (x, z) \rightleftharpoons (y, z)
\]
Lindley, Wadler and Yallop (2008), proved the equivalences

\[
\text{Idiom} = \text{Arrow} + (x \rightsquigarrow y \cong 1 \rightsquigarrow (x \rightarrow y)),
\]

\[
\text{Monad} = \text{Arrow} + (x \rightsquigarrow y \cong x \rightarrow (1 \rightsquigarrow y))
\]
Lindley, Wadler and Yallop (2008), proved the equivalences

\[
\text{Idiom} = \text{Arrow} + (x \rightsquigarrow y \simeq 1 \rightsquigarrow (x \rightarrow y)),
\]
\[
\text{Monad} = \text{Arrow} + (x \rightsquigarrow y \simeq x \rightarrow (1 \rightsquigarrow y))
\]

Following a syntactic approach: calculi and translations.
Lindley, Wadler and Yallop (2008), proved the equivalences

\[
\text{Idiom} = \text{Arrow} + (x \rightsquigarrow y \cong 1 \rightsquigarrow (x \to y)),
\]
\[
\text{Monad} = \text{Arrow} + (x \rightsquigarrow y \cong x \to (1 \rightsquigarrow y))
\]

Following a syntactic approach: calculi and translations.

We aim for a semantic explanation, modelling:

- Arrows as profunctors $F^{\text{op}} \times F \to S$ with monoid structure.
- Monads and idioms as functors $F \to S$ with monoid structure.
Monads, idioms and arrows have

- an operation embedding pure values: \textit{return}, \textit{pure} and \textit{arr}.
- an operation sequencing computations: \textit{>>=}, \textit{⊛} and \textit{≫}.
Monads, idioms and arrows have

- an operation embedding pure values: \textit{return}, \textit{pure} and \textit{arr}.
- an operation sequencing computations: \(\Rightarrow\), \(\odot\) and \(\gg\).

Resemble monoids.
Monads, idioms and arrows have

- an operation embedding pure values: \( \text{return} \), \( \text{pure} \) and \( \text{arr} \).
- an operation sequencing computations: \( (\gg) \), \( (\DOTimes) \) and \( (\gg) \).

Resemble monoids.

We model computational effects using *monoidal categories*.

\[
\text{Monad} \Rightarrow \text{Monoid in } ([F, S], \circ)
\]

\[
\text{Idiom} \Rightarrow \text{Monoid in } ([F, S], \star)
\]

\[
\text{Arrow} \Rightarrow \text{Monoid in } ([F^{\text{op}} \times F, S], \otimes)
\]
The category of finitary endofunctors $[\mathcal{F}, \mathcal{S}]$ has a substitution monoidal structure.

$$(F \circ G)X = \int^Y FY \times (Y \to GX)$$

The inclusion $i : \mathcal{F} \to \mathcal{S}$ acts as unit.
The category of finitary endofunctors $[\mathbb{F}, \mathbb{S}]$ has a substitution monoidal structure.

$$(F \circ G)X = \int^Y FY \times (Y \to GX)$$

The inclusion $i : \mathbb{F} \to \mathbb{S}$ acts as unit.

A monoid

$$i \xrightarrow{\text{return}} M \xleftarrow{(\gg)} M \circ M$$

in $([\mathbb{F}, \mathbb{S}], \circ, i)$ is a monad.
Monoidal structures: ⋆

The category $[\mathcal{F}, \mathcal{S}]$ also has a *convolution* monoidal structure.

$$(F \ast G)X = \int^Y FY \times G(Y \to X)$$

The inclusion $i : \mathcal{F} \to \mathcal{S}$ also acts as the unit.
Monoidal structures: ⋆

The category $\mathbb{[F, S]}$ also has a convolution monoidal structure.

$$(F \ast G)X = \int^Y FY \times G(Y \to X)$$

The inclusion $i : F \to S$ also acts as the unit.

A monoid

$$i \xrightarrow{\text{pure}} F \xleftarrow{\ast} F \ast F$$

in $\mathbb{[F, S]}$, $\ast$, $i$ is an idiom.
Intermezzo: strong profunctors

Profunctors compatible with the underlying cartesian structure.

Definition: strong profunctor

A profunctor $P : F^{op} \times F \to S$ is strong if it comes equipped with a family of morphisms $\text{str}_{X, Y, Z} : P(X, Y) \to P(X \times Z, Y \times Z)$ natural in $X, Y$ and dinatural in $Z$ such that the following equations hold:

$P(id, \pi_1) \circ \text{str}_{X, Y, 1} = P(\pi_1, id)$,

$\text{str}_{X, Y, W} \circ \text{str}_{X, Y, V} = P(\alpha^{-1}, \alpha) \circ \text{str}_{X, Y, V \times W}$.
Intermezzo: strong profunctors

Profunctors compatible with the underlying cartesian structure.

**Definition: strong profunctor**

A profunctor $P : F^{\text{op}} \times F \to S$ is *strong* if it comes equipped with a family of morphisms

$$\text{str}_{X,Y,Z} : P(X, Y) \to P(X \times Z, Y \times Z)$$

natural in $X$, $Y$ and dinatural in $Z$ such that the following equations hold:

$$P(\text{id}, \pi_1) \circ \text{str}_{X,Y,1} = P(\pi_1, \text{id}),$$

$$\text{str}_{X,Y,W} \circ \text{str}_{X,Y,V} = P(\alpha^{-1}, \alpha) \circ \text{str}_{X,Y,V \times W}$$
Monoidal structures: \( \otimes \)

Strong profunctors \( \mathcal{F}^{\text{op}} \times \mathcal{F} \to \mathcal{S} \) have composition of profunctors.

\[
(P \otimes Q)(X, Y) = \int^W P(X, W) \times Q(W, Y)
\]

The hom-set \( \text{Hom}_{\mathcal{F}} : \mathcal{F}^{\text{op}} \times \mathcal{F} \to \mathcal{S} \) as the unit.
Monoidal structures: \( \otimes \)

Strong profunctors \( F^{\text{op}} \times F \to S \) have composition of profunctors.

\[
(P \otimes Q)(X, Y) = \int^W P(X, W) \times Q(W, Y)
\]

The hom-set \( \text{Hom}_F : F^{\text{op}} \times F \to S \) as the unit.

A monoid

\[
\text{Hom}_F \xrightarrow{\text{arr}} A \xleftarrow{\text{(\(\otimes\))}} A \otimes A
\]

in \( ([F^{\text{op}} \times F, S], \otimes, \text{Hom}_F) \) is an arrow.
The equations II

\[
\text{Idiom} = \text{Arrow} + (x \leadsto y \cong 1 \leadsto (x \rightarrow y)),
\]

\[
\text{Monad} = \text{Arrow} + (x \leadsto y \cong x \rightarrow (1 \leadsto y))
\]

We have defined \textit{Idiom}, \textit{Monad} and \textit{Arrow} in our model:

\[
\text{Monad} \Rightarrow \text{Monoid in } ([F, S], \circ)
\]

\[
\text{Idiom} \Rightarrow \text{Monoid in } ([F, S], \ast)
\]

\[
\text{Arrow} \Rightarrow \text{Monoid in } ([F^{\text{op}} \times F, S]_S, \otimes)
\]

Isomorphisms on the right still missing.
The equations II

Idiom = Arrow + (x ⇀ y ≅ 1 ⇀ (x → y)),
Monad = Arrow + (x ⇀ y ≅ x → (1 ⇀ y))

We have defined Idiom, Monad and Arrow in our model:

Monad ⇒ Monoid in ([F, S], ○)
Idiom ⇒ Monoid in ([F, S], ⋆)
Arrow ⇒ Monoid in ([F^{op} × F, S]_s, ⊗)

Isomorphisms on the right still missing.
As a first step, we model the isomorphisms for profunctors. If $A$ is the strong profunctor underlying the arrow ($\leadsto$)

\[
x \leadsto y \cong 1 \leadsto (x \to y) \quad \Rightarrow \quad A(x, y) \cong A(1, x \to y),
\]
\[
x \leadsto y \cong x \to (1 \leadsto y) \quad \Rightarrow \quad A(x, y) \cong ix \to A(1, y).
\]
Formalising the isomorphisms

As a first step, we model the isomorphisms for profunctors. If $A$ is the strong profunctor underlying the arrow $(\sim\sim)$

\[
x \sim \sim y \cong 1 \sim \sim (x \to y) \quad \Rightarrow \quad A(x, y) \cong A(1, x \to y),
\]

\[
x \sim \sim y \cong x \to (1 \sim \sim y) \quad \Rightarrow \quad A(x, y) \cong ix \to A(1, y).
\]

We try to factorise

\[
A(1, x \to y) \quad \text{and} \quad ix \to A(1, y)
\]

as functors applied to $A$ on $x$ and $y$. 
A strong profunctor in $[\text{F}^{\text{op}} \times \text{F}, \mathbb{S}]_s$ can be mapped to a functor $\text{F} \to \mathbb{S}$ by evaluating its first parameter.
A strong profunctor in \([F^{op} \times F, S]_S\) can be mapped to a functor \(F \to S\) by evaluating its first parameter.

In particular, evaluating with 1, we obtain

\[
\begin{align*}
-^* : [F^{op} \times F, S]_S & \longrightarrow [F, S] \\
A^* &= Z \mapsto A(1, Z) \\
\tau^* Z &= \tau_{1,Z}
\end{align*}
\]
The functor \(-^*\) has left and right adjoints:
The functor -∗ has left and right adjoints:

\[-! : [F, S] \longrightarrow [F^{op} \times F, S]_s\]

\[F! = (X, Y) \mapsto F(X \to Y)\]
The functor \(-\ast\) has left and right adjoints:

\[-! : \mathbb{F}, \mathbb{S} \rightarrow \mathbb{F}^{\text{op}} \times \mathbb{F}, \mathbb{S}_s\]

\[F_! = (X, Y) \mapsto F(X \rightarrow Y)\]

\[-\ast : \mathbb{F}, \mathbb{S} \rightarrow \mathbb{F}^{\text{op}} \times \mathbb{F}, \mathbb{S}_s\]

\[F_\ast = (X, Y) \mapsto i \ X \rightarrow F \ Y\]
The functor -$^*$ has left and right adjoints:

\[-! : [F, S] \rightarrow \text{[}F^{\text{op}} \times F, S\text{]}_s\]

\[F! = (X, Y) \mapsto F(X \rightarrow Y)\]

\[-*: [F, S] \rightarrow \text{[}F^{\text{op}} \times F, S\text{]}_s\]

\[F* = (X, Y) \mapsto i X \rightarrow F Y\]

We end up with an adjoint triple

\[-! \dashv -^* \dashv -^*\]
We obtain the diagram

\[
[F, S] \leftarrow \downarrow \ast \rightarrow [F^{\text{op}} \times F, S]_s
\]
We obtain the diagram

\[
\begin{array}{c}
[F, S] & \xrightarrow{-!} & [F^{\text{op}} \times F, S]_s \\
\downarrow_{-\ast} & & \downarrow_{-\ast} \\
\end{array}
\]

and the isomorphisms become

\[
A(x, y) \cong A(1, x \to y) \implies A \cong (A^*)_! \\
A(x, y) \cong ix \to A(1, y) \implies A \cong (A^*)_*
\]
What about the monoidal structures?

Idiom = Arrow + \((x \rightsquigarrow y \cong 1 \rightsquigarrow (x \rightarrow y))\),

Monad = Arrow + \((x \rightsquigarrow y \cong x \rightarrow (1 \rightsquigarrow y))\).
What about the monoidal structures?

Idiom = Arrow + \((x \rightsquigarrow y \cong 1 \rightsquigarrow (x \to y))\),

Monad = Arrow + \((x \rightsquigarrow y \cong x \to (1 \rightsquigarrow y))\).

On the isomorphisms we only dealt with the objects.
What about the monoidal structures?

Idiom = \text{Arrow} + (x \rightsquigarrow y \cong 1 \rightsquigarrow (x \rightarrow y)),

Monad = \text{Arrow} + (x \rightsquigarrow y \cong x \rightarrow (1 \rightsquigarrow y)).

On the isomorphisms we only dealt with the objects.

**Theorem**

For an adjoint triple $F \dashv G \dashv H$, we have that the comonad $FG$ and the monad $HG$ are adjoint $FG \dashv HG$.
What about the monoidal structures?

Idiom = Arrow + \((x \leadsto y \cong 1 \leadsto (x \to y))\),

Monad = Arrow + \((x \leadsto y \cong x \to (1 \leadsto y))\).

On the isomorphisms we only dealt with the objects.

**Theorem**

For an adjoint triple \(F \dashv G \dashv H\), we have that the comonad \(FG\) and the monad \(HG\) are adjoint \(FG \dashv HG\).

From the adjoint triple

\[
\dashv \quad \dashv \quad \dashv
\]

we obtain

\[
(-^*)! = \Box \dashv \diamond = (-^*)_\ast
\]
In our case, the comonad □ and the monad ◊ are idempotent.
In our case, the comonad □ and the monad ◊ are idempotent.

**Definition: T-monoid**

If $T : C \rightarrow C$ is an idempotent (co)monad, then a $T$-monoid is quadruple $(M, m, e, \alpha)$ where

- $(M, m : M \otimes M \rightarrow M, e : I \rightarrow M)$ is a monoid;
- $(M, \alpha : TM \rightarrow M)$ is a $T$-algebra.

$T$-monoids form a category $\text{Mon}(T)$. 
In our case, the comonad □ and the monad ◊ are idempotent.

**Definition: T-monoid**

If $T : C \to C$ is an idempotent (co)monad, then a $T$-monoid is a quadruple $(M, m, e, \alpha)$ where

- $(M, m : M \otimes M \to M, e : I \to M)$ is a monoid;
- $(M, \alpha : TM \to M)$ is a T-algebra.

$T$-monoids form a category $\text{Mon}(T)$.

For idempotent (co)monads, (co)algebras are isos. A $T$-monoid $(M, m, e, \alpha)$ is a Monoid on $C + (M \cong TM)$
The equivalences

\[
\text{Idiom} = \text{Arrow} + (x \rightsquigarrow y \cong 1 \rightsquigarrow (x \to y))
\]

\[
\downarrow
\]

Mon ([F, S]) and Mon (□) are equivalent categories.
The equivalences

\[ \text{Idiom} = \text{Arrow} + (x \rightsquigarrow y \cong 1 \rightsquigarrow (x \to y)) \]

\[ \Downarrow \]

\[ \text{Mon} ([\mathbb{F}, \mathbb{S}]) \text{ and } \text{Mon (□)} \text{ are equivalent categories.} \]

\[ \text{Monad} = \text{Arrow} + (x \rightsquigarrow y \cong x \to (1 \rightsquigarrow y)) \]

\[ \Downarrow \]

\[ \text{Mon} ([\mathbb{F}, \mathbb{S}]) \text{ and } \text{Mon (◊)} \text{ are equivalent categories.} \]
To prove

\[ \text{Mon}(\mathbb{F}, S) \text{ and } \text{Mon}(\Box) \text{ are equivalent categories} \]
Proof sketch 1

To prove

\[ \text{Mon}([F, S]) \text{ and } \text{Mon}(\square) \text{ are equivalent categories} \]

note that both functors are monoidal (monoidal adjunction)

\[
\begin{array}{c}
(F, S, \star) \\
\downarrow \Downarrow \\
(F^{\text{op}} \times F, S, \otimes)
\end{array}
\]
To prove

\[ \text{Mon}([F, S]) \text{ and Mon}(\square) \text{ are equivalent categories} \]

note that both functors are monoidal (monoidal adjunction)

\[ ([F, S], \ast) \quad \perp \quad ([F^{\text{op}} \times F, S], \otimes) \]

Functors lift to categories of monoids.
Proof sketch II

In the case

\[ \text{Mon}([\mathcal{F}, \mathcal{S}]) \text{ and Mon}(\diamond) \text{ are equivalent categories} \]
Proof sketch II

In the case

\[ \text{Mon}([F, S]) \text{ and Mon}(\diamond) \text{ are equivalent categories} \]

the adjunction

\[
\begin{array}{c}
([F^{\text{op}} \times F, S], \otimes) \\
\downarrow
\end{array}
\begin{array}{c}
\text{perp}
\end{array}
\begin{array}{c}
([F, S], \circ)
\end{array}
\]

is a monoidal conjunction. No guarantees that \(-^*\) preserves monoids.
Proof sketch II

In the case

\[ \text{Mon}([F, S]) \text{ and Mon}(\Diamond) \] are equivalent categories

the adjunction

\[
\begin{array}{ccc}
([F^{\text{op}} \times F, S]_S, \otimes) & \bot & ([F, S], \circ) \\
\end{array}
\]

is a monoidal conjunction. No guarantees that \(-\ast\) preserves monoids.

A result by Porst and Street gives conditions when an opmonoidal functor preserves monoids.
Conclusions

We have extended the notions of computation as monoids view to show a semantic counterpart to Lindley et al.’s result.
Conclusions

We have extended the notions of computation as monoids view to show a semantic counterpart to Lindley et al.’s result.

Further work includes

- replacing $F$ and $S$.
- relating to relative monads and other solutions that do not suffer of size issues.
- seeing how comonads and other notions fit in the picture.