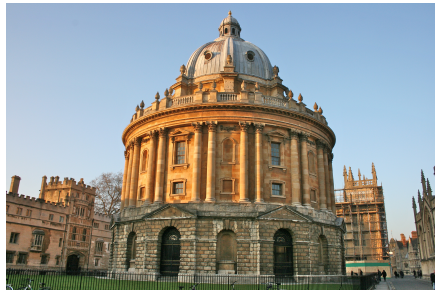
Mathematical Foundations of Bidirectional Transformations

Michael Johnson

Optus Macquarie Cyber Security Hub
Macquarie University Sydney

BXSS, Oxford, July 25-29, 2016



History

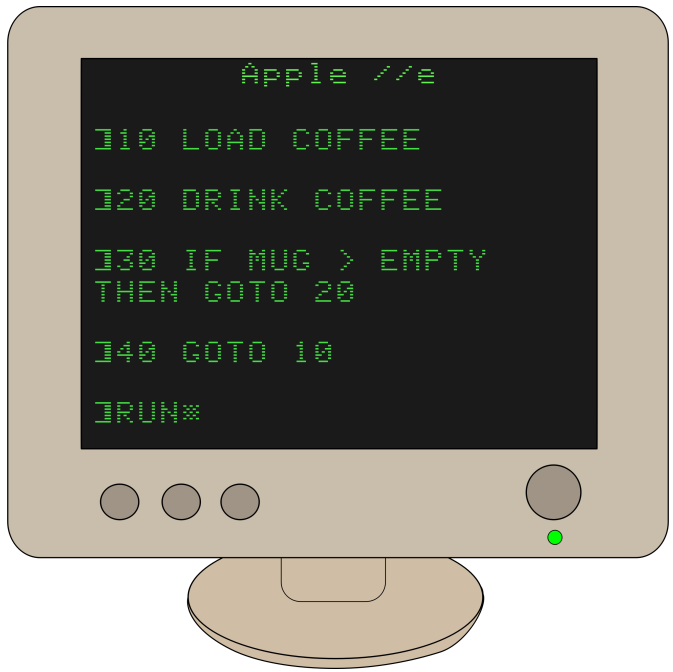
1986

History

1986 — Just communications and just computations

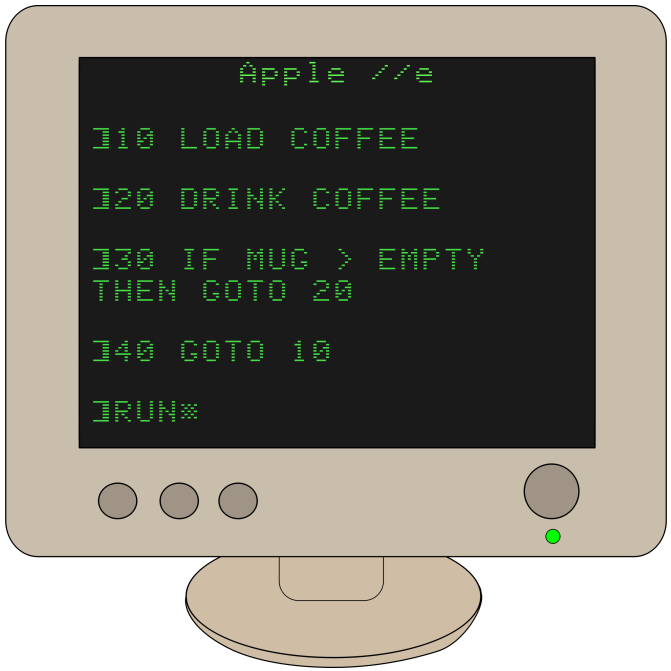
History

1986 — Just communications and just computations



History

1986 — Just communications and just computations



History

1986 — Just communications and just computations

1996

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

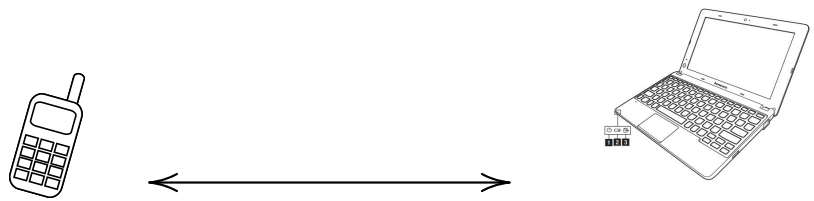
History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization



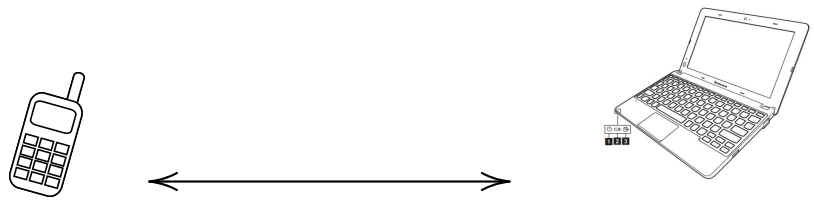
History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization



A bidirectional transformation

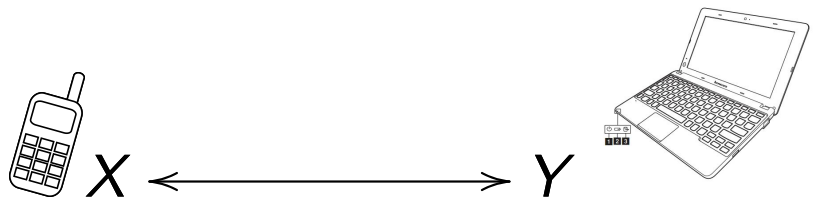
History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization



A bidirectional transformation

(1) A *consistency relation* R between the possible states X and the possible states Y

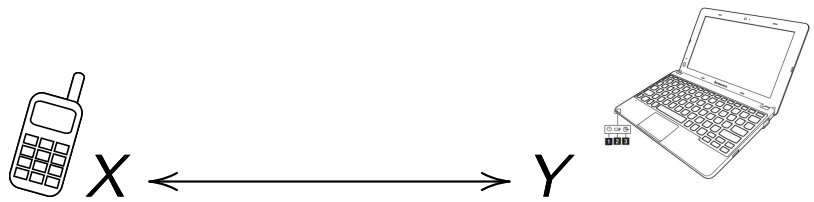
History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization



A bidirectional transformation

(1) A *consistency relation* R between the possible states X and the possible states Y, so $R \subset X \times Y$

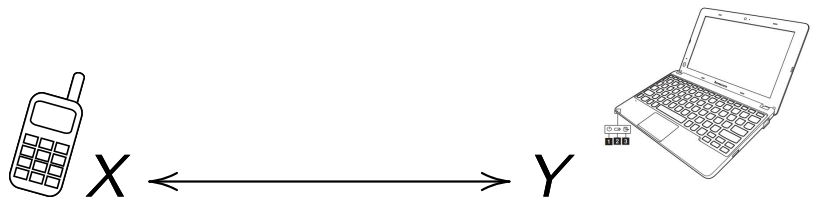
History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization



A bidirectional transformation

(1) A *consistency relation* R between the possible states X and the possible states Y, together with (2) *consistency restorers*.

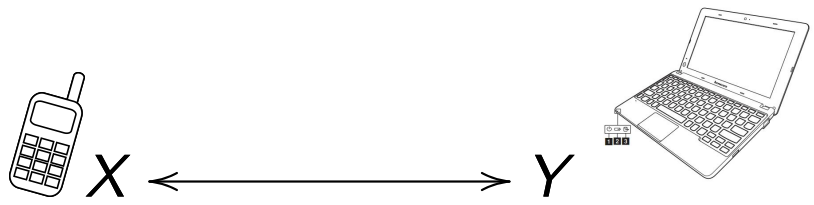
History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization



A bidirectional transformation

A *consistency relation* R between the possible states X and the possible states Y, together with *consistency restorers*

$$\triangleleft : X \times Y \longrightarrow X \quad \text{and} \quad \triangleright : X \times Y \longrightarrow Y$$

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011

History

1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011

History

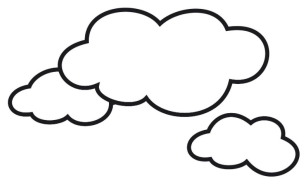
1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011



History

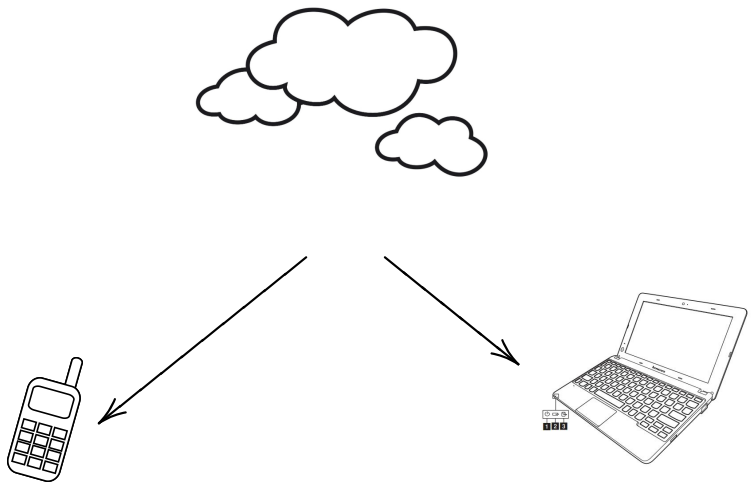
1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011



History

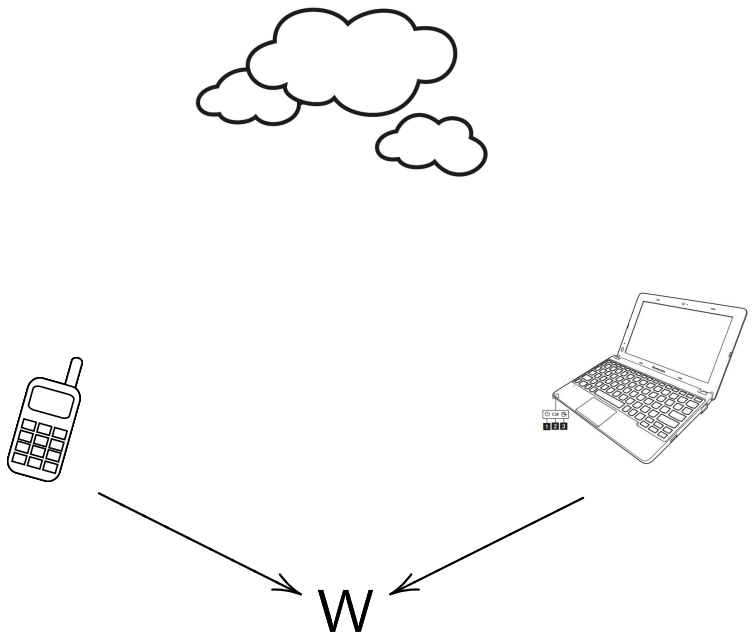
1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011



History

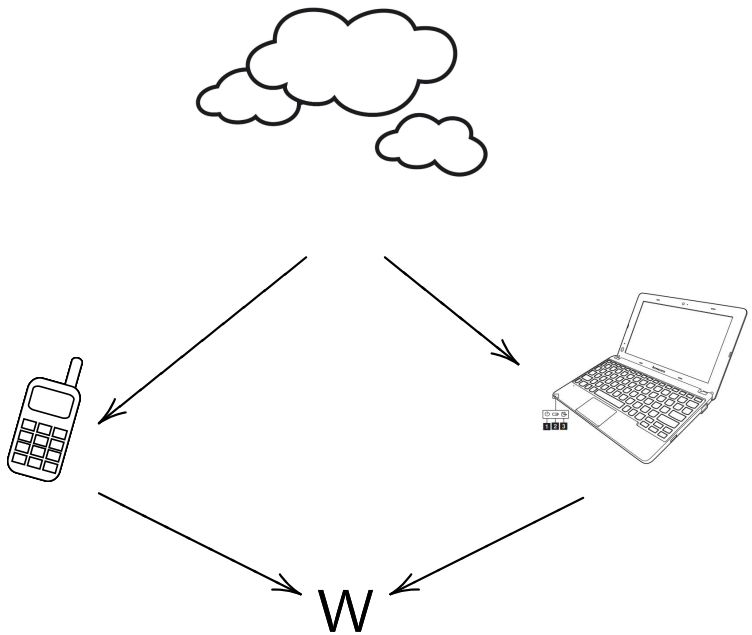
1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011



History

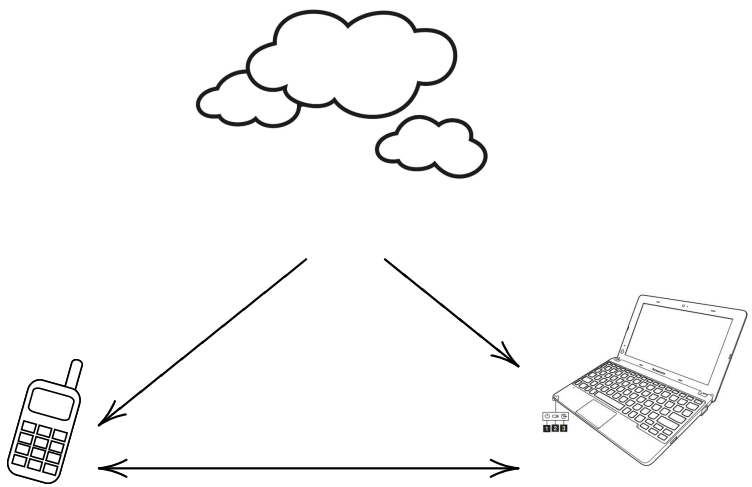
1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

2006 — Synchronization

2011 — The great big system in the sky



History

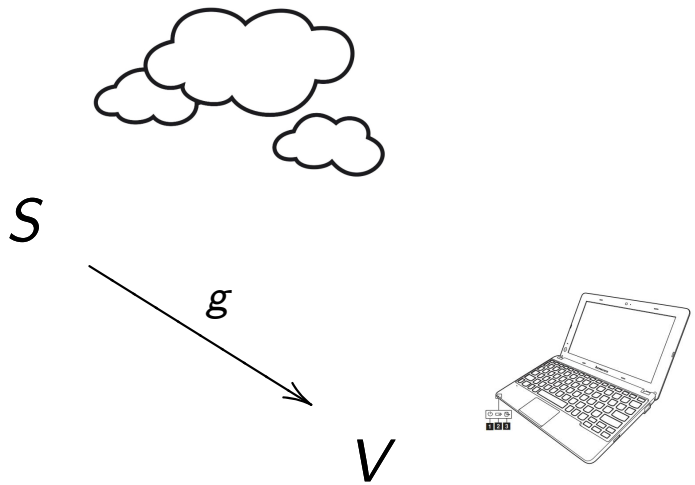
1986 — Just communications and just computations

1996 — Web: Ordinary people's computers communicate

2001 — Convergence: Phones manage data too

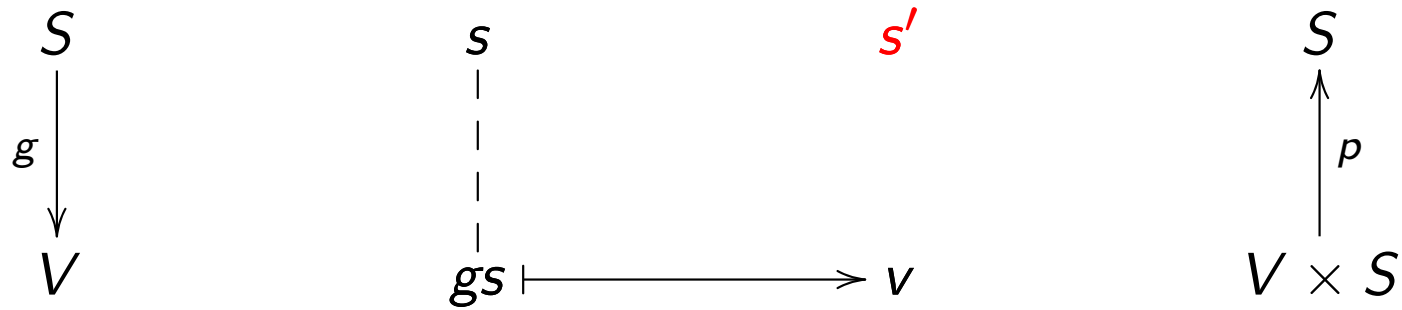
2006 — Synchronization

2011 — The great big system in the sky



Lens

(Asymmetric, set-based: S and V are sets, g and p are functions, and $s, s' \in S$ and $v, v' \in V$.)

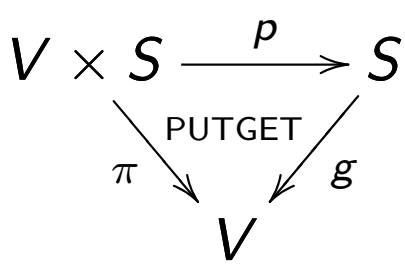


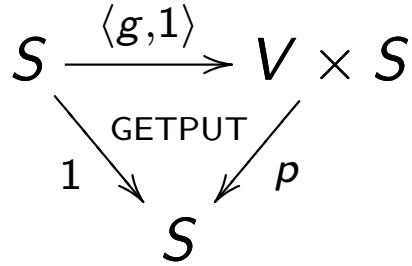
Satisfying axioms

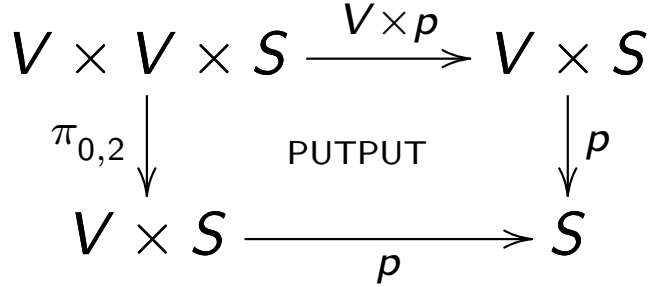
PutGet: $g(p(v, s)) = v$
GetPut: $p(g(s), s)) = s$ and
PutPut: $p(v', p(v, s)) = p(v', s)$.

(Notice that the arrow $gs \longrightarrow v$ is not used.)

The axioms diagramatically





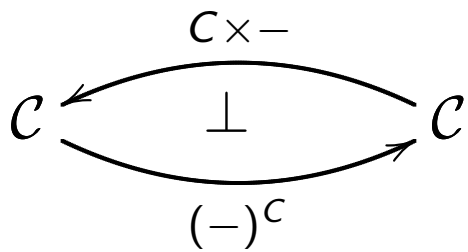


Adjoints (Functional Programming)

You're most likely familiar with

$$\frac{(C \times A \longrightarrow B)}{(A \longrightarrow B^C)}$$

which comes from



and adjoints come with a unit (boring) and a counit (important)

$$\varepsilon : C \times X^C \longrightarrow X$$

Adjoints (Algebra)

Now think of free monoids (free groups, free ...)

$$\frac{(FA \longrightarrow B)}{(A \longrightarrow UB)}$$

which comes from (F free, U underlying)

$$\mathcal{A}lg \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\perp}}}} \mathcal{S}et$$

and adjoints come with a unit

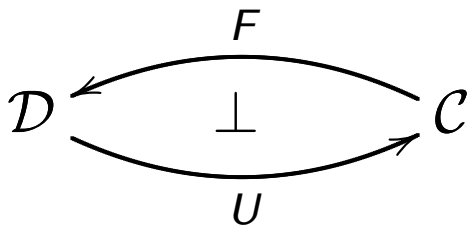
$$\eta_A : A \longrightarrow UFA$$

and a counit

$$\varepsilon_B : FUB \longrightarrow B$$

Monads (Universal Algebra)

Any adjunction



gives rise to an endofunctor $T = UF : \mathcal{C} \longrightarrow \mathcal{C}$ with units

$$\eta_A : A \longrightarrow UFA$$

amounting to a natural transformation $\eta : 1_{\mathcal{C}} \longrightarrow UF$.
Furthermore, TT reduces to T (terms of terms are terms) by a natural transformation $\mu : T^2 \longrightarrow T$ (setting $\mu = U\varepsilon F$).

So ...

Monad (definition)

A monad is one of the three standard ways of presenting (specifying) a universal algebra (a class of algebras).

A *monad* consists of an endofunctor $T : \mathcal{C} \longrightarrow \mathcal{C}$ together with natural transformations

$$\mu : T^2 \longrightarrow T,$$

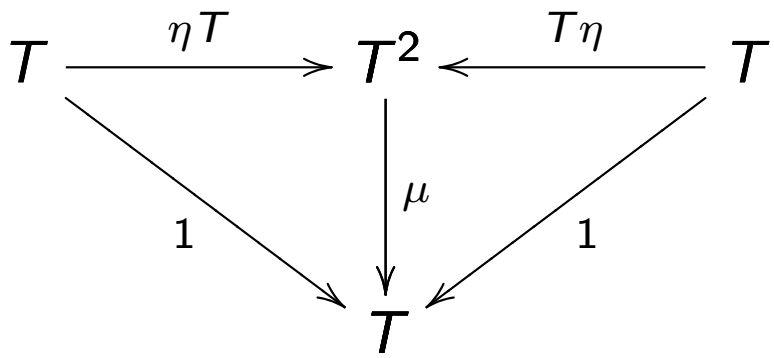
called the multiplication, and

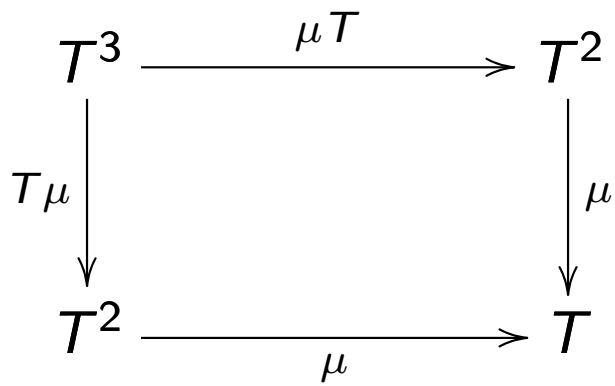
$$\eta : 1_{\mathcal{C}} \longrightarrow T,$$

called the unit, satisfying...

Monad axioms

(Remember: $T : \mathcal{C} \longrightarrow \mathcal{C}$, $\mu : T^2 \longrightarrow T$ and $\eta : 1_{\mathcal{C}} \longrightarrow T$)



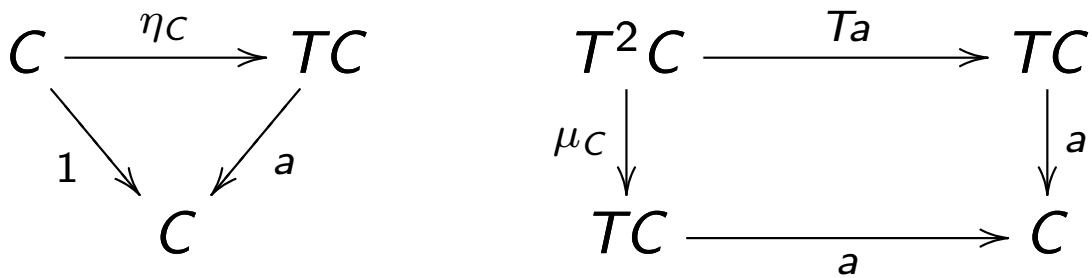


called left-ientity, right-identity and associativity, respectively.

Algebras

Monads are just ways of forming and managing terms. What we really work with (even if you're a functional programmer) is algebras.

An *algebra* for the monad $T : \mathcal{C} \longrightarrow \mathcal{C}$ is an object C of $\mathcal{C}$ along with a morphism $a : TC \longrightarrow C$ of $\mathcal{C}$, called the *action*, rendering commutative



These are (also) called the identity and associativity laws respectively (but for the algebra now).

Monadicity

Suppose we have an adjunction $F \dashv U$.
Are the algebras for the monad UF the ones you might expect?
Usually, yes. When they are, we say that the right adjoint U is *monadic*.

But sometimes, no. For example, Categories are not monadic over the category of sets. This isn't really surprising: The data for building a free category shouldn't be just a set of objects, or a set of arrows, or even a pair of sets of objects and arrows. The basic (non-categorical) structure needs to be pre-specified.
Categories are indeed monadic over the category of directed (multi-) graphs.

The moral here is that we need to be thoughtful about what is the appropriate base category for our monads.

Defining categories

Now might be a good time to give a brief formal definition of a category.

A *category* is a

- directed graph whose nodes are called *objects* and edges are called *arrows*, together with
- a specified composition which is associative and has identities.

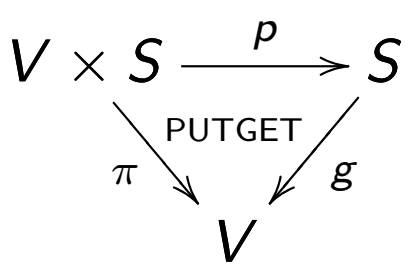
(A pair of arrows is *composable* if and only if they line up head to tail: $A \longrightarrow B \longrightarrow C$.)

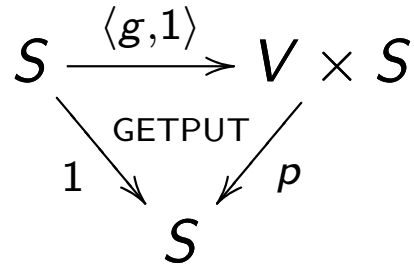
Time for a short pause

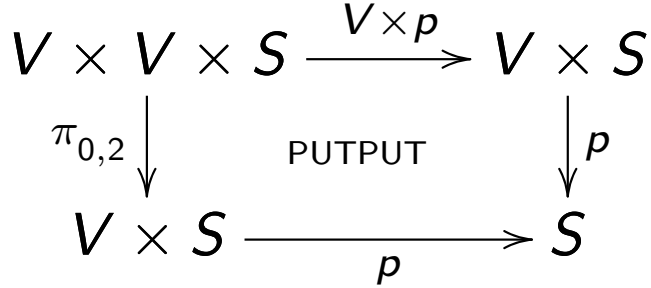
And congratulate yourself: Very few people learn about adjunctions and monadicity before defining categories!

Let's compare

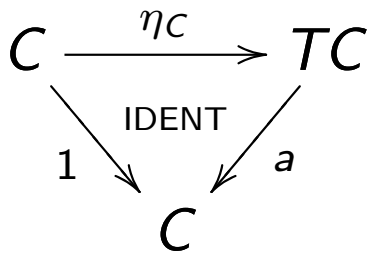
Lens axioms

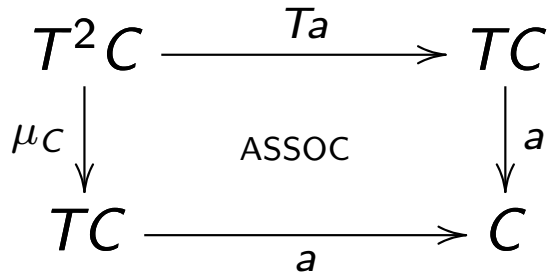






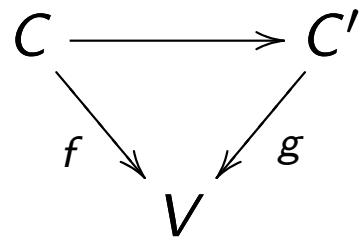
Monad algebra axioms





Slice category

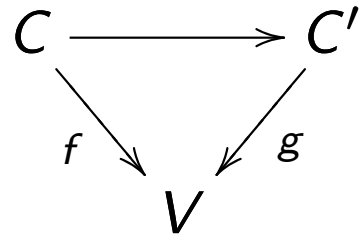
Given a category $\mathcal{C}$ and an object V in $\mathcal{C}$, the *slice category* $\mathcal{C}/V$ has as objects arrows of $\mathcal{C}$ with codomain V and as arrows from $f : C \longrightarrow V$ to $g : C' \longrightarrow V$ arrows of $\mathcal{C}$ from $C \longrightarrow C'$ making the triangle



commute.

Slice category

Given a category $\mathcal{C}$ and an object V in $\mathcal{C}$, the *slice category* $\mathcal{C}/V$ has as objects arrows of $\mathcal{C}$ with codomain V and as arrows from $f : C \longrightarrow V$ to $g : C' \longrightarrow V$ arrows of $\mathcal{C}$ from $C \longrightarrow C'$ making the triangle



commute.

For example, if **set** is the category of sets, and V is a set (for example the set of states of a system) then **set**$/V$ denotes a slice category whose objects one might think of as potential gets for lenses.

Lenses are algebras

Note that $R = V \times -$, fibred by the projection, is a monad on $\mathbf{set}/V$.

Lenses are algebras

Note that $R = V \times -$, fibred by the projection, is a monad on $\mathbf{set}/V$.

Theorem

An algebra for the monad R is a (set-based asymmetric) lens satisfying the three lens axioms (PutGet, GetPut and PutPut).

More precisely, $R = \Delta\Sigma$ where $\Sigma : \mathbf{set}/V \longrightarrow \mathbf{set}$ is the projection from the slice category ("sum up the fibres" to get just a set) and $\Delta : \mathbf{set} \longrightarrow \mathbf{set}/V$ takes X to $\Delta X = \pi : V \times X \longrightarrow V$ (called "fibred over V by the projection of the product").
Furthermore $\Sigma \dashv \Delta$ and, provided V is non-empty, Δ is monadic.

Constant complement

Corollary

Every set-based asymmetric lens satisfying the three lens axioms is isomorphic to one of the form $g = \pi : V \times C \longrightarrow V$ *with* p *given by the constant complement updating strategy*

$$p(v, s) = p(v, (v', c)) = (v, c).$$

Constant complement

Corollary

Every set-based asymmetric lens satisfying the three lens axioms is isomorphic to one of the form $g = \pi : V \times C \longrightarrow V$ *with* p *given by the constant complement updating strategy*

$$p(v, s) = p(v, (v', c)) = (v, c).$$

This is important and valuable (for example, the updating strategy in SQL is entirely based on this), but not very interesting and quite restricted.

For some time people wanting more interesting lenses said "let's give up PutPut — it looked overly strong anyway" but life is much more interesting than that(!).

Stay tuned...

End of Monday lecture

Would you like some lunch?