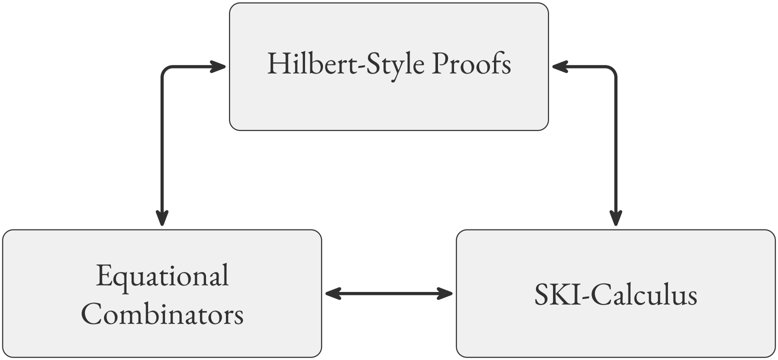
Proofs as Combinators

A Simple Desultory Philippic

Florian Guthmann

2025-01-09

$Id: talk.tex,v 1.9 2025/01/09 15:16:16 flo Exp $


Hilbert-Style Proofs
Equational Combinators
SKI-Calculus

Part I

Hilbert-Style Proofs

Axiom Schemas or Inference Rules?

Axiom Schemas

Inference Rules

The Implicational Fragment

$$\frac{\Gamma \vdash \varphi \Rightarrow \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\ \text{MP}$$

$$\frac{}{\Gamma \vdash (\varphi \Rightarrow \psi \Rightarrow \vartheta) \Rightarrow (\varphi \Rightarrow \psi) \Rightarrow \varphi \Rightarrow \vartheta}\ \text{S}$$

$$\frac{}{\Gamma \vdash \varphi \Rightarrow \psi \Rightarrow \varphi}\ \text{K}$$

Identity

$$\dfrac{\dfrac{\dfrac{}{\mathsf{Q} \Rightarrow \mathsf{P} \Rightarrow \varphi \Rightarrow \varphi}\,\mathrm{S} \qquad \dfrac{}{\varphi \Rightarrow (\psi \Rightarrow \varphi) \Rightarrow \varphi}\,\mathrm{K}}{\mathsf{P} \Rightarrow \varphi \Rightarrow \varphi}\,\mathrm{MP} \qquad \dfrac{}{\varphi \Rightarrow \psi \Rightarrow \varphi}\,\mathrm{K}}{\varphi \Rightarrow \varphi}\,\mathrm{MP}$$

$$\mathsf{Q} := \varphi \Rightarrow (\psi \Rightarrow \varphi) \Rightarrow \varphi$$

$$\mathsf{P} := \varphi \Rightarrow \psi \Rightarrow \varphi$$

tbd

Theorem (Deduction)

$$\Gamma, \varphi \vdash \psi \iff \Gamma \vdash \varphi \Rightarrow \psi$$

Adding more Axioms

$$\frac{}{\Gamma \vdash \bot \Rightarrow \varphi}$$

$$\frac{}{\Gamma \vdash \varphi \Rightarrow \varphi \vee \psi} \qquad \frac{}{\Gamma \vdash \psi \Rightarrow \varphi \vee \psi}$$

$$\frac{}{\Gamma \vdash (\varphi \Rightarrow \vartheta) \Rightarrow (\psi \Rightarrow \vartheta) \Rightarrow \varphi \vee \psi \Rightarrow \vartheta}$$

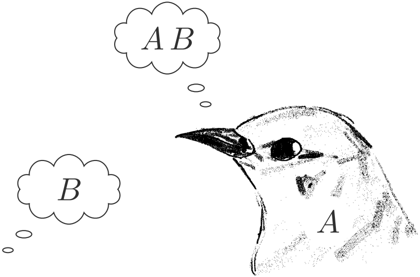
$$\frac{}{\Gamma \vdash \varphi \wedge \psi \Rightarrow \varphi} \qquad \frac{}{\Gamma \vdash \varphi \wedge \psi \Rightarrow \psi}$$

$$\frac{}{\Gamma \vdash (\vartheta \Rightarrow \varphi) \Rightarrow (\vartheta \Rightarrow \psi) \Rightarrow \vartheta \Rightarrow \varphi \wedge \psi}$$

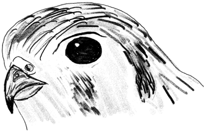
Part II

To Mock a Mockingbird

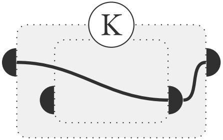
Forests



Kestrel

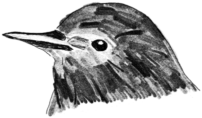


$$\mathrm{K}\, x\, y = x$$

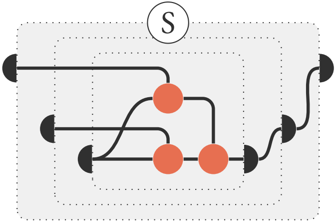


`const`

Starling



$$S\,x\,y\,z = x\,z\,(y\,z)$$

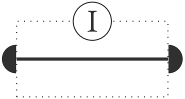



```
\x y z -> x z (y z)
```
[1]

[1] More on this later

Identity Bird

$$\mathrm{I}\, x = x$$



`id`

Identity Defined

Hilbert Style

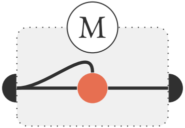
$$\dfrac{\dfrac{\dfrac{}{\mathsf{Q} \Rightarrow \mathsf{P} \Rightarrow \varphi \Rightarrow \varphi}\,\mathrm{S} \qquad \dfrac{}{\varphi \Rightarrow (\psi \Rightarrow \varphi) \Rightarrow \varphi}\,\mathrm{K}}{\mathsf{P} \Rightarrow \varphi \Rightarrow \varphi}\,\mathrm{MP} \qquad \dfrac{}{\varphi \Rightarrow \psi \Rightarrow \varphi}\,\mathrm{K}}{\varphi \Rightarrow \varphi}\,\mathrm{MP}$$

$$\mathrm{I} = (\mathrm{S}\,\mathrm{K})\,\mathrm{K} = \mathrm{S}\,\mathrm{K}\,\mathrm{K}$$

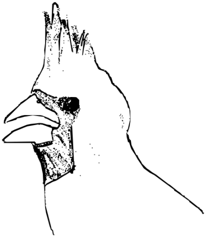
Mockingbird

$$\mathrm{M}\,x = x\,x$$

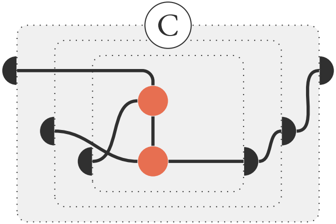
$$\mathrm{M} = \mathrm{S\,I\,I}$$



Cardinal

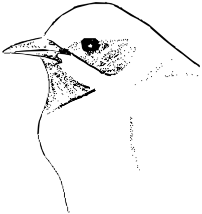


$$\mathrm{C}\,x\,y\,z = x\,z\,y$$

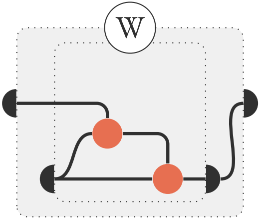


`flip`

Warbler



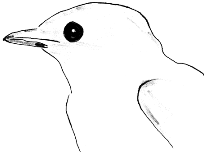
$$\mathrm{W}\, x\, y = x\, y\, y$$



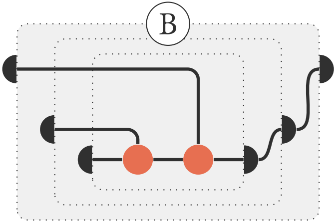

```
\x y -> x y y
```
[2]

[2] Again, more on this later

Bluebird



$$\mathrm{B}\,x\,y\,z = x\,(y\,z)$$

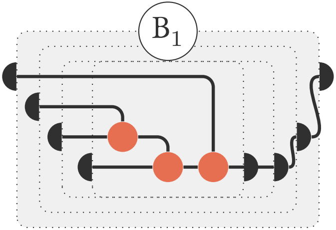



```
(.)
```

Blackbird

$$\mathrm{B}_1\, x\, y\, z\, m = x\,(y\, z\, m)$$

$$\mathrm{B}_1 = \mathrm{B}\,\mathrm{B}\,\mathrm{B}$$



The Beatles

The BCKW-Forest

The forest containing a *Bluebird*, *Cardinal*, *Kestrel* and *Warbler*.

Each bird represents a unique capability:

B	parenthesising
C	reordering
K	discarding
W	duplication

⇝ interesting subforests!

The BCKW-Forest

The forest containing a *Bluebird*, *Cardinal*, *Kestrel* and *Warbler*.

Each bird represents a unique capability:

B	parenthesising
C	reordering
K	discarding
W	duplication

$\rightsquigarrow$ interesting subforests!

BCK $\leftrightsquigarrow$ a fragment of affine logic

BC $\leftrightsquigarrow$ a fragment of linear logic

The SK-Forest

The forest containing a *Starling* and *Kestrel*.

S represents multiple capabilities at once:

S	parenthesising, reordering, duplication
K	discarding

Comparison

	SK	BCKW
S	S	B (B W) (B B C)
B	S (K S) K	B
C	S (B B S) (K K)	C
K	K	K
W	S S (K I)	W

Part III

Combinatory Logic

Syntax

$$F, G ::= v \mid \mathrm{K} \mid \mathrm{S} \mid F\,G$$

where $v \in \mathcal{V}$

Small-Step Operational Semantics

$$\frac{}{\mathrm{K}\,F\,G \to F}\ \mathrm{K}$$

$$\frac{}{\mathrm{S}\,F\,G\,H \to F\,H\,(G\,H)}\ \mathrm{S}$$

$$\frac{F \to F'}{F\,G \to F'\,G}\ \mathrm{app_l} \qquad \frac{G \to G'}{F\,G \to F\,G'}\ \mathrm{app_r}$$

Weak equality

Let $=_{\mathrm{w}}$ be the least equivalence relation containing $\to$:

$$\frac{}{F =_{\mathrm{w}} F} \qquad \frac{G =_{\mathrm{w}} F}{F =_{\mathrm{w}} G}$$

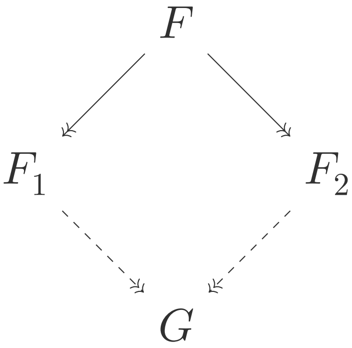
$$\frac{F \to G}{F =_{\mathrm{w}} G} \qquad \frac{G \to F}{F =_{\mathrm{w}} G}$$

Multi Step Relation

Let $\twoheadrightarrow$ be the reflexive transitive closure of $\rightarrow$:

$$\frac{}{F \twoheadrightarrow F} \qquad\qquad \frac{F \rightarrow G \qquad G \twoheadrightarrow H}{F \twoheadrightarrow H}$$

Church-Rosser



Combinators as λ-terms

$$
\begin{aligned}
&(-)_\Lambda : \mathrm{CL} \to \Lambda \\
&(v)_\Lambda = v \\
&(\mathrm{K})_\Lambda = \lambda x.\, \lambda y.\, x \\
&(\mathrm{S})_\Lambda = \lambda x.\, \lambda y.\, \lambda z.\, x\, z\, (y\, z) \\
&(F\, G)_\Lambda = (F)_\Lambda\, (G)_\Lambda
\end{aligned}
$$

λ-terms as Combinators?

$$
\begin{aligned}
(-)_{\mathrm{CL}} &: \Lambda \to \mathrm{CL} \\
(v)_{\mathrm{CL}} &= v \\
(s\,t)_{\mathrm{CL}} &= (s)_{\mathrm{CL}}\,(t)_{\mathrm{CL}} \\
(\lambda v.\,t)_{\mathrm{CL}} &= \;?
\end{aligned}
$$

Combinatory Abstraction

$$\lambda^\star(-).\,(-) : \mathcal{V} \times \mathrm{CL} \to \mathrm{CL}$$

$$\lambda^\star v.\, x = \begin{cases} \mathrm{I} & v = x \\ \mathrm{K}\, x & \text{otherwise} \end{cases}$$

$$\lambda^\star v.\, \mathrm{K} = \mathrm{K}\,\mathrm{K}$$

$$\lambda^\star v.\, \mathrm{S} = \mathrm{K}\,\mathrm{S}$$

$$\lambda^\star v.\, (B\,C) = \mathrm{S}\,(\lambda^\star v.\, B)\,(\lambda^\star v.\, C)$$

$\lambda^\star$ is β-reductive

Theorem

$$(\lambda^\star x.\, A\, B) \twoheadrightarrow (A\ [B/x])$$

Typing Relation (à la Curry)

$$\frac{}{\Gamma \vdash \mathrm{K} \colon \sigma \to \tau \to \sigma}\ \text{K}$$

$$\frac{}{\Gamma \vdash \mathrm{S} \colon (\sigma \to \tau \to \varrho) \to (\sigma \to \tau) \to \sigma \to \varrho}\ \text{S}$$

$$\frac{\Gamma \vdash M \colon \sigma \to \tau \qquad \Gamma \vdash N \colon \sigma}{\Gamma \vdash M\, N \colon \tau}\ \text{MP}$$

Church-Style Typing

Reminder
In the λ-calculus, church style typing is done by annotating λ-abstractions with the type of the bound variable.

Instead, index S and K with the choices of types:

$$F, G ::= v \mid \mathrm{K}_{\sigma,\tau} \mid \mathrm{S}_{\sigma,\tau,\varrho} \mid F\,G$$

where $\sigma, \tau, \varrho \in \text{Type}$

Curry-Howard

Hilbert-Style	Typed Combinatory Logic
Modus Ponens	Application
Assumption	Variable
Axiom Schemes	K and S

Part IV

Golfing, for Fun and Profit

Combinators in Haskell

Remember `S = \x y z -> x z (y z)` ?

Surely we shouldn't be churning butter with a toothpick like that!

Combinators in Haskell

Remember `S = \x y z -> x z (y z)` ?

Surely we shouldn't be churning butter with a toothpick like that!

The `Prelude` comes to our rescue!

Applicative Functors[3][4]

An endofunctor $F\colon \mathcal{C} \to \mathcal{C}$ is called *applicative*, if there exist

- $\mathrm{pure}\colon A \to FA$
- $\circledast\colon F(B^A) \times FA \to FB$

such that

$$
\begin{aligned}
\mathrm{pure}(\mathrm{id}_A) \circledast Fa &= a \\
\mathrm{pure}(\circ) \circledast u \circledast v \circledast w &= u \circledast (v \circledast w) \\
\mathrm{pure}(f) \circledast \mathrm{pure}(a) &= \mathrm{pure}(f(a)) \\
u \circledast \mathrm{pure}(a) &= \mathrm{pure}(\lambda f.\, f(a)) \circledast u
\end{aligned}
$$

[3] with a grain of salt

[4] Mcbride and Paterson, "Applicative programming with effects".

Environment Functor is Applicative

$$(-)^A : \mathcal{C} \to \mathcal{C}$$

$$\begin{array}{ccc} X & \longmapsto & X^A \\ \downarrow f & & \downarrow g \mapsto f \circ g \\ Y & \longmapsto & Y^A \end{array}$$

$$\text{pure} : X \to X^A$$

$$\circledast : Y^{X^A} \times X^A \to Y^A$$

Environment Functor is Applicative

$$(-)^A \colon \mathcal{C} \to \mathcal{C}$$

$$\begin{array}{ccc} X & \longmapsto & X^A \\ \Big\downarrow f & & \Big\downarrow g \mapsto f \circ g \\ Y & \longmapsto & Y^A \end{array}$$

$$\text{pure} \colon X \to A \to X$$
$$\text{pure} = \mathrm{K}$$
$$\circledast \colon Y^{X^A} \times X^A \to Y^A$$

Environment Functor is Applicative

$$(-)^A \colon \mathcal{C} \to \mathcal{C}$$

$$\begin{array}{ccc} X & \longmapsto & X^A \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle g \mapsto f \circ g} \\ Y & \longmapsto & Y^A \end{array}$$

$$\text{pure} \colon X \to A \to X$$

$$\text{pure} = \mathrm{K}$$

$$\circledast \colon (A \to X \to Y) \to (A \to X) \to A \to Y$$

$$\circledast = \mathrm{S}$$

Combinators in Haskell

```haskell
K = pure
```

```haskell
S = (<*>)
```

```haskell
B = pure (<*>) <*> pure
```

```haskell
C = ((.) . (<*>)) <*> pure pure
```

```haskell
W = (<*>) <*> pure id = join
```

- Keenan, David C. *To Dissect a Mockingbird: A Graphical Notation for the Lambda Calculus with Animated Reduction*. Apr. 2014. URL: https://dkeenan.com/Lambda.
- Mcbride, Conor and Ross Paterson. “Applicative programming with effects”. In: *J. Funct. Program.* 18.1 (Jan. 2008), pp. 1–13. ISSN: 0956-7968. DOI: 10.1017/S0956796807006326. URL: https://doi.org/10.1017/S0956796807006326.
- Schönfinkel, Moses. “Über die Bausteine der mathematischen Logik”. In: *Mathematische Annalen* 92.3 (1924), pp. 305–316.
- Smullyan, Raymond M. *To Mock a Mockingbird*. Knopf, 1985.
- Sørensen, Morten Heine. *Lectures on the Curry-Howard Isomorphism*. Ed. by PawełUrzyczyn. Boston: Elsevier, 2006.

Monoidal Categories

A *monoidal* category is a category $\mathcal{C}$ with a bifunctor $\otimes \colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and an object $1 \in \mathcal{C}$ such that there exist natural isomorphisms

- $\alpha_{A,B,C} \colon A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$
- $\rho_A \colon A \otimes 1 \cong A$
- $\lambda_B \colon 1 \otimes B \cong B$

Lax Monoidal Endofunctors

Let $(\mathcal{C}, \otimes, 1)$ be a monoidal category[5]. A *lax monoidal* endofunctor on $\mathcal{C}$ is a functor $F : \mathcal{C} \to \mathcal{C}$ with

- a morphism $1 \to F1$
- a natural transformation $\mu_{X,Y} : FX \otimes FY \to F(X \otimes Y)$

such that

$$
\begin{array}{ccc}
(FA \otimes FB) \otimes FC & \xrightarrow{\alpha_{FA,FB,FC}} & FA \otimes (FB \otimes FC) \\
\downarrow{\scriptstyle \mu_{A,B} \otimes \mathrm{id}} & & \downarrow{\scriptstyle \mathrm{id} \otimes \mu_{B,C}} \\
F(A \otimes B) \otimes FC & & FA \otimes F(B \otimes C) \\
\downarrow{\scriptstyle \mu_{A \otimes B, C}} & & \downarrow{\scriptstyle \mu_{A, B \otimes C}} \\
F((A \otimes B) \otimes C) & \xrightarrow{F(\alpha_{A,B,C})} & F(A \otimes (B \otimes C))
\end{array}
$$

[5] e.g. $(\mathsf{Set}, \times, \{\bullet\})$

Lax Monoidal Endofunctors

Let $(\mathcal{C}, \otimes, 1)$ be a monoidal category[5]. A *lax monoidal* endofunctor on $\mathcal{C}$ is a functor $F : \mathcal{C} \to \mathcal{C}$ with

- a morphism $1 \to F1$
- a natural transformation $\mu_{X,Y} : FX \otimes FY \to F(X \otimes Y)$

such that

$$
\begin{array}{ccc}
FA \otimes 1 & \xrightarrow{\mathrm{id} \otimes \eta} & FA \otimes F1 \\
\downarrow \rho_A & & \downarrow \mu_{A,1} \\
FA & \xleftarrow[F(\rho_A)]{} & F(A \otimes 1)
\end{array}
\qquad
\begin{array}{ccc}
1 \otimes FB & \xrightarrow{\eta \otimes \mathrm{id}} & F1 \otimes FB \\
\downarrow \lambda_B & & \downarrow \mu_{1,B} \\
FB & \xleftarrow[F(\lambda_B)]{} & F(1 \otimes B)
\end{array}
$$

[5] e.g. $(\mathsf{Set}, \times, \{\bullet\})$

Applicative Functors

Let $(\mathcal{C}, \otimes, 1)$ be a monoidal category. A lax monoidal functor $F : \mathcal{C} \to \mathcal{C}$ is called *strong* if there is a natural transformation

$$\tau_{A,B} : A \otimes FB \to F(A \otimes B)$$

such that