Kleene Monads in a Long While

Sergey Goncharov
FAU Erlangen-Nürnberg

February 14, 2023, Oberseminar Theoretische Informatik
**Kleene iteration** is iteration of regular expressions, e.g. 
\((0 \lor 1)^* \cdot 0 \cdot (0 \lor 1)^*\) – all binary strings that contain 0 at least once

**Kleene algebra** is a lightweight equational theory of (Kleene) iteration, complete over formal languages

It is extremely popular and has lots of extensions: hybrid, concurrent, stateful, etc

**Kleene monads** is a simple categorification of Kleene algebras

**Elgot monads** provide a vastly more general notion of iteration, and are highly compositional

- But not quite that popular 😞

What is precisely the delta between Kleene and Elgot?

Can we bridge mathematical (mental, social, psychological, . . .) gap between them?

💡 We approach by formalising a theory of while-loops
Recalling Kleene algebra

Recalling Kleene monads

Recalling Elgot monads

While-monads

Relating Kleene and Elgot

Conclusions
Kleene Algebra for (Kleene) Iteration
A **Kleene algebra** is a structure \((S, \bot, \eta, \lor, ;, (-)^*)\), where

- \((S, \bot, \eta, \lor, ;)\) is an idempotent semiring:
  - \((S, \bot, \lor)\) is a **commutative** \((x \lor y = y \lor x)\) and **idempotent** \((x \lor x = x)\) monoid
  - \((S, \eta, ;)\) is a monoid
  - **distributive laws:**
    \[
    \begin{align*}
    x; (y \lor z) &= x; y \lor x; z \\
    (x \lor y); z &= x; z \lor y; z \\
    x; \bot &= \bot \\
    \bot; x &= \bot
    \end{align*}
    \]
    (thus, \(S\) is partially ordered: \(x \leq y\) iff \(x \lor y = y\))

- **Kleene iteration** satisfies \(x^* = \eta \lor x; x^*\), and

\[
\begin{align*}
\frac{x; y \lor z \leq y}{x^*; z \leq y} & \quad \frac{x \lor z; y \leq z}{x; y^* \leq z}
\end{align*}
\]

Equivalently: \(x^*; z\) is a least fixpoint of \(x; (-) \lor z\) and \(z; y^*\) is a least fixpoint of \((-); y \lor z\)

**Intuition:** \(\bot\) is a deadlock, \(\eta\) is a neutral program, ; is sequential composition, \(\lor\) is non-deterministic choice
Regular expressions

Algebraic language of finite state machines and beyond

Relational semantics of programs

Relational reasoning and verification, e.g. via dynamic logic

Plenty of extensions:

- modal $\Rightarrow$ modal Kleene algebra (Struth et al.)
- stateful $\Rightarrow$ KAT + B! (Grathwohl, Kozen, Mamouras)
- concurrent $\Rightarrow$ concurrent Kleene algebra (Hoare et al.)
- nominal $\Rightarrow$ nominal Kleene algebra (Kozen et al.)
- with differential equations $\Rightarrow$ differential dynamic logic (Platzer et al.)
- etc., etc., etc.

decidability and completeness properties (most famously w.r.t. formal languages and relational interpretations)
A minimalist extension is **Kleene algebra with tests (KAT)**, which adds

- another Kleene algebra $B$ of **tests**
- an operation-preserving inclusion $?: B \hookrightarrow S$
- complementation operator $\overline{(-)}: B \rightarrow B$, such that

\[
\overline{a} \lor a = \top \quad \overline{\overline{a}} = a \quad \overline{a \lor b} = \overline{a}; \overline{b} \quad \bot = \eta
\]

(this makes $B$ into a Boolean algebra)

This enables encodings

- **Branching** $(\text{if } b \text{ then } p \text{ else } q)$ as $b?; p \lor \overline{b}?; q$
- **Looping** $(\text{while } b \text{ do } p)$ as $(b?; p)^*; \overline{b}$
- **Hoare triples** $\{a\} p \{b\}$ as $a?; p; b? = a?; p$

In particular, we can embed deterministic semantics to non-deterministic semantics
Nondeterminism is a computational effect. If we want to add other effects, we need to add more operations and equations

**Example:** Probabilistic choice \((+p : S \rightarrow S)_{p \in [0,1]}\), plus axioms of barycentric algebras

\[
x +_0 y = y \quad x +_p x = x \quad x +_p y = x +_{1-p} y
\]

\[
(x +_p y) +_q z = x +_{p+q-pq} (y +_p q-pq z)
\]

plus **distributivity** \((x \lor y) +_p z = (x +_p z) \lor (y +_p z)\)

**Problem 1:** Not all effects mix well with Kleene algebra, e.g. exception raising:

\[
\text{raise } e_1 = \text{raise } e_1; \bot = \text{raise } e_2; \bot = \text{raise } e_2
\]

**Problem 2:** We might want a stronger theory of non-determinism, e.g. the law \(x; (y \lor z) = x; y \lor x; z\) is unsound w.r.t. strong bisimilarity

**Problem 3:** What is a disciplined way to extend iteration to other settings (muti-type, multi-effect, foundation-independent)?
CATEGORIZING ITERATION
Definition (Monad)

A **monad** $T$ (on a category $C$) is given by a **Kleisli triple** $(T, \eta, -^*)$ where

- $T : |C| \to |C|$
- $\eta$ is a family of morphisms $\eta_X : X \to TX$, forming **Monad unit**
- $(-)^*$ assigns to each $f : X \to TY$ a morphism $f^* : TX \to TY$

satisfying the laws: $\eta^* = \text{id}$, $f^* \eta = f$, $(f^* g)^* = f^* g^*$

This entails that

- $T$ is a functor, $\eta$ is a natural transformation
- $f : X \to TY$ and $g : Y \to TZ$ (regarded as programs) can be **Kleisli composed** to $f; g = g \cdot f = g^* f : X \to TZ$

By varying $T$ we obtain various ‘**generalized programs**’ $f : X \to TY$ while programs of the form $\eta f$ can be seen as ‘**pure programs**’

**Example:** $T = \text{powerset} \Rightarrow$ generalized programs = non-deterministic programs, pure programs = deterministic programs = functions
**Definition (Kleene-Kozen Category)**

Call $\mathcal{C}$ a **Kleene-Kozen category** if it is enriched over join-semilattices with $\bot$ and strict join-preserving morphisms, and there is **Kleene iteration**

$$(-)^* : \text{Hom}(X, X) \to \text{Hom}(X, X),$$

such that, given $f : Y \to Y$, $g : Y \to Z$ and $h : X \to Y$, $gf^*$ is the least fixpoint of $g \lor (-)f$ and $f^*h$ is the least fixpoint of $h \lor f(-)$.

**Proposition:** A Kleene algebra is a Kleene-Kozen single-object category

**Definition:**\(^1\) A monad $T$ is a **Kleene monad** if its Kleisli category is Kleene-Kozen.

This suggests a canonical definition of an $n$-sorted Kleene algebra as a Kleene-Kozen category with $n$ objects.

E.g. KAT becomes a certain two-sorted category.

\(^1\)Goncharov, Schröder, and Mossakowski, “Kleene monads: handling iteration in a framework of generic effects”, 2009.
Examples:

- For any monoid $M$, $\mathcal{P}(M \times -)$ is a Kleene monad (generalizes language-theoretic models)
- If $T$ is a Kleene monad then so is the state-monad transform $(T(- \times S))^S$ for any $S$ (generalized relational models)

Non-Examples:

- Maybe monad $TX = X + 1$ has $\bot$ but no $\top$
- $TX = \mathcal{P}^+(X + 1)$ (where $\mathcal{P}^+$ is non-empty powerset) fails $f \lor \bot = f$
- $TX = \mathcal{P}(A^* \times X + A^\omega)$ fails $f; \bot = \bot$
- $TX$ is a final coalgebra for $\mathcal{P}_{\omega_1}(X + A \times (-))$ (where $\mathcal{P}_{\omega_1}$ is the countable powerset functor) fails $f; (g \lor h) = f; g \lor f; h$
Definition (Elgot monad)

A (complete) Elgot monad\(^2\) in a category with binary coproducts is a monad \(T\) equipped with an Elgot iteration operator

\[
(-)^\dagger : \text{Hom}(X, T(Y + X)) \to \text{Hom}(X, TY),
\]

satisfying four laws: fixpoint, uniformity, naturality and codiagonal

- Elgotness is robust and stable under many monad transformers
  - \( T \mapsto T(M \times -) \) (writer)
  - \( T \mapsto T(- + E) \) (exception)
  - \( T \mapsto (T(- \times S))^S \) (state)
  - \( T \mapsto \nu\gamma. T(- + H\gamma) \) (resumption)

- Laws go back to Elgot\(^3\), except for uniformity

- All previous examples are Elgot

---


\(^3\)Elgot, “Monadic Computation And Iterative Algebraic Theories”, 1975.
**Fixpoint** \( (f : X \rightarrow T(Y + X)) : \)

\[
\begin{align*}
\begin{array}{c}
\text{Fixpoint} \quad (f : X \rightarrow T(Y + X)) : \\
\end{array}
\end{align*}
\]

**Uniformity** \( (f : X \rightarrow T(Z + X), g : Y \rightarrow T(Z + Y), h : X \rightarrow Y) : \)

\[
\begin{align*}
\begin{array}{c}
\text{Uniformity} \quad (f : X \rightarrow T(Z + X), g : Y \rightarrow T(Z + Y), h : X \rightarrow Y) : \\
\end{array}
\end{align*}
\]
**Naturality** \((f: X \to T(Y + X), g: Y \to TZ)\):

\[
\begin{align*}
  \xrightarrow{\text{Naturality}} \quad & \quad f \quad g \quad X \quad Y \quad Z \\
  & \quad \quad x \quad \quad \quad x \quad \quad \quad y \quad \quad \quad z
\end{align*}
\]

\[
\begin{align*}
  = \quad & \quad f \quad g \quad X \quad Y \quad Z \\
  & \quad \quad x \quad \quad \quad x \quad \quad \quad y \quad \quad \quad z
\end{align*}
\]

**Codiagonal** \((f: X \to T(Y + (X + X)))\):

\[
\begin{align*}
  \xrightarrow{\text{Codiagonal}} \quad & \quad f \quad y \quad X \\
  & \quad \quad x \quad \quad \quad y
\end{align*}
\]

\[
\begin{align*}
  = \quad & \quad f \quad y \quad X \\
  & \quad \quad x \quad \quad \quad y
\end{align*}
\]

**Naturality** and **Codiagonal** are basically coherence laws
Theorem

Every Kleene monad is an Elgot monad under

\[ \begin{align*}
    x & \quad f & \quad x \\
    y & \quad x & \quad f
\end{align*} \]

where \( \rightarrow \) is the deadlock

Proof.

Essentially a known fact about iteration theories\(^4\)

But, how large is the gap between Kleene monads and Elgot monads + \( \{\bot, \vee\} \)?

---

is derivable\(^5\), also it was included in the original axiomatization

Q: So, maybe there are more derivable axioms?

A: In fact, it is provable that the present axiomatization is minimal

While-Monads
A key distinction between Elgot iteration and Kleene iteration is that the former needs a (simple) type system, while the later can make do without any types whatsoever.

Kleene algebra with tests has two sorts, and yet no types

**Definition (Decisions)**

We call a family \((\mathcal{C}^d(X) \subseteq \mathcal{C}(X, X + X))_{X \in \mathcal{C}}\) in a category \(\mathcal{C}\), decisions if every \(\mathcal{C}^d(X)\) contains \text{inl}, \text{inr}, and \(\mathcal{C}^d\) is closed under if-then-else.

We encode logical operations on decisions as follows:

- \(\text{ff} = \text{inl}\), \(b \& b = \text{if } b \text{ then } c \text{ else } \text{ff}\), \(~b = \text{if } b \text{ then } \text{ff} \text{ else } \text{tt}\),
- \(\text{ff} = \text{inr}\), \(b || c = \text{if } b \text{ then } \text{tt} \text{ else } c\).

By definition, decisions can range from the smallest family with \(\mathcal{C}^d(X) = \{\text{ff}, \text{tt}\}\), to the greatest one with \(\mathcal{C}^d(X) = \mathcal{C}(X, X + X)\).
A while-monad is a monad $T$, equipped with an operator

$$
\text{while} : C^d_T(X) \times C(X, TX) \rightarrow C(X, TX),
$$

such that the following axioms are satisfied

**W-Fix** \quad \text{while } b \ p = \text{if } b \text{ then } p; (\text{while } b \ p) \text{ else } \eta

**W-Or** \quad \text{while } (b \ || \ c) \ p = (\text{while } b \ p); \text{while } c \ (p; \text{while } b \ p)

**W-And** \quad \frac{\eta h; b = \eta u; \text{ff}}{\text{while } (b \ &\& (c \ || \ \eta u; \text{ff})) \ p = \text{while } b \ (\text{if } c \text{ then } p \text{ else } \eta h)}

**W-Uni** \quad \frac{\eta h; b = \text{if } c \text{ then } \eta h'; \text{tt} \text{ else } \eta u; \text{ff} \quad \eta h'; p = q; \eta h}{\eta h; \text{while } b \ p = (\text{while } c \ q); \eta u}
While-Monads and Elgot Monads

- From $(-)^\dagger$ to while: while $b \ p = (\text{if } b \text{ then } p; \text{tt else } \text{ff})^\dagger$, diagrammatically, while $b \ p$ is expressed as

- From while to $(-)^\dagger$

Theorem

If for all $X, Y \in |C|$, $\eta(\text{inl} + \text{inr}) \in C^d_T(X + Y)$ then $T$ is an Elgot monad iff it is a while-monad w.r.t. $C^d_T$
Theorem

A monad $T$ is a Kleene monad iff

- the Kleisli category of $T$ is enriched over join-semilattices with $\bot$ and strict join-preserving morphisms;
- there is an operator $(-)^*: \text{Hom}(X, TX) \to \text{Hom}(X, TX)$, such that
  
  1. $f^* = \eta \lor f^* \cdot f$
  2. $\eta^* = \eta$
  3. $f^* = (f \lor \eta)^*$
  4. $f \cdot h = g \cdot f \Rightarrow f \cdot h^* = g^* \cdot f$

Corollary: Since Kleene algebra is a special case, this is also a complete axiomatization for Kleene algebra
Theorem

A monad $T$ is a Kleene monad iff

1. $T$ is Elgot
2. the Kleisli category of $T$ is enriched over bounded join-semilattices and strict join-preserving morphisms
3. $T$ satisfies the law $(\eta \text{ inl} \lor \eta \text{ inr})^\dagger = \eta$
4. $T$ satisfies strong uniformity:

\[
\begin{align*}
\begin{array}{c}
\begin{tikzpicture}
  \node (x) at (0,0) [circle,draw] {$x$};
  \node (h) at (1,0) [rectangle,draw] {$h$};
  \node (g) at (2,0) [rectangle,draw] {$g$};
  \node (y) at (2,0.5) [circle,draw] {$y$};
  \node (z) at (2,-0.5) [circle,draw] {$z$};
  \draw (x) edge (h) (h) edge (g) (g) edge [loop above] (g) (g) edge (y) (y) edge (z) (x) edge (z);
\end{tikzpicture}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\begin{tikzpicture}
  \node (x) at (0,0) [circle,draw] {$x$};
  \node (h) at (1,0) [rectangle,draw] {$h$};
  \node (y) at (2,0.5) [circle,draw] {$y$};
  \node (z) at (2,-0.5) [circle,draw] {$z$};
  \draw (x) edge (h) (h) edge (y) (y) edge (z) (x) edge (z);
\end{tikzpicture}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\begin{tikzpicture}
  \node (x) at (0,0) [circle,draw] {$x$};
  \node (h) at (1,0) [rectangle,draw] {$h$};
  \node (g) at (2,0) [rectangle,draw] {$g$};
  \node (y) at (2,0.5) [circle,draw] {$y$};
  \node (z) at (2,-0.5) [circle,draw] {$z$};
  \draw (x) edge [loop below] (x) (x) edge (h) (h) edge (g) (g) edge [loop above] (g) (g) edge (y) (y) edge (z) (x) edge (z);
\end{tikzpicture}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\begin{tikzpicture}
  \node (x) at (0,0) [circle,draw] {$x$};
  \node (h) at (1,0) [rectangle,draw] {$h$};
  \node (y) at (2,0) [circle,draw] {$y$};
  \node (z) at (2,0) [circle,draw] {$z$};
  \draw (x) edge [loop below] (x) (x) edge (h) (h) edge (y) (y) edge (z) (x) edge (z);
\end{tikzpicture}
\end{array}
\end{align*}
\]

where $h$ is strict $h; \delta = \delta$, and where $\delta = (\eta \text{ inr})^\dagger$ by definition
Elgot monads that fail strong uniformity without the strictness assumption on $h$ are easy to manufacture

- For example, $\mathcal{P}(- +1)$ is an Elgot monad, obtained by transforming $\mathcal{P}$
- Besides $\delta = \{\text{inr }\star\}$ it contains another ‘divergence’ $\perp = \{\}$
- The premise of strong uniformity is satisfied with $h = \perp$ and $f = \eta \text{ inr}$, but the conclusion $\delta = (\eta \text{ inr})^\dagger = g^\dagger$ generally fails

It is thus always reasonable to restrict $h$ in strong uniformity to strict programs, i.e. $h; \delta = \delta$

If we add finite nondeterminism and other laws, it is more difficult to construct a separating example, however

**Proposition**

There exists an Elgot monad $\mathcal{T}$, whose Kleisli category is enriched over bounded semi-lattices, $(\eta \text{ inl } \lor \eta \text{ inr})^\dagger = \eta$, but $\mathcal{T}$ fails strong uniformity.
Proof Sketch

1. The idea is based on Kozen’s separating example for left-handed and right-handed Kleene algebras\(^6\)

2. Consider the submonad \(T\) of the **continuation monad** (neighbourhood monad, dualization monad) \((\mathbb{N} \to 2) \to 2\), formed by those \(f\) that preserve finite unions:

\[
f\{\}\ = \{\} \quad f(A \cup B) = f(A) \cup f(B)
\]

3. Using the fact that every \(TX\) is a complete lattice, define \(f^\dagger\) as a least fixpoint, using the **Knaster-Tarski theorem**. Hence \((\eta \ \text{inl} \lor \eta \ \text{inr})^\dagger = \eta\)

4. Enrichment in semilattices follow by definition

5. Remaining Elgot monad laws follow by **transfinite induction**

6. If \(T\) was a Kleene monad, any \(\text{Hom}(X, TX)\) would be a Kleene algebra, but Kozen showed it is not \(\Rightarrow T\) is not Kleene

---

So, what is the fundamental version of uniformity?

- Bloom and Ésik\(^7\) argued that there is only one theory of iteration, accepting only those instances of uniformity, where premises are internally provable. This is insufficient for practical equational reasoning.
- Uniformity is arguably the most conservative extension of the purely equational theory and is a part of the Elgot monad axiomatization.
- Strong uniformity with strict maps bridges the gap with Kleene monads.

We obtained a spectrum

\[ x \lor \bot = x \]

Elgot monads \[ x \lor x = x \]

\[ \lor, \bot \]

\[ x; (y \lor z) = x; y \lor x; z \]

Kleene monads

**Further Work:**

- How can we define while-algebras, generalizing KAT, in the upshot?
  - The main hurdle is uniformity, which should preferably not allude to a yet another sort of programs, and be properly weaker than strong uniformity

- When can we equivalently replace while \( + \lor \) with Kleene star \( + \lor \) so that \( p^* = \text{while} (\text{ff } \lor \text{tt}) \ p \)?

- Generic completeness theorems, generalizing completeness for KAT