Kleene Iteration: From Kleene Algebra Onwards

Sergey Goncharov

School of Computer Science, Univ. Birmingham

MGS Christmas seminar, Univ. Sheffield, 17 December, 2024

Kleene Iteration: From Kleene Algebra Onwards

Sergey Goncharov

School of Computer Science, Univ. Birmingham

MGS Christmas seminar, Univ. Sheffield, 17 December, 2024

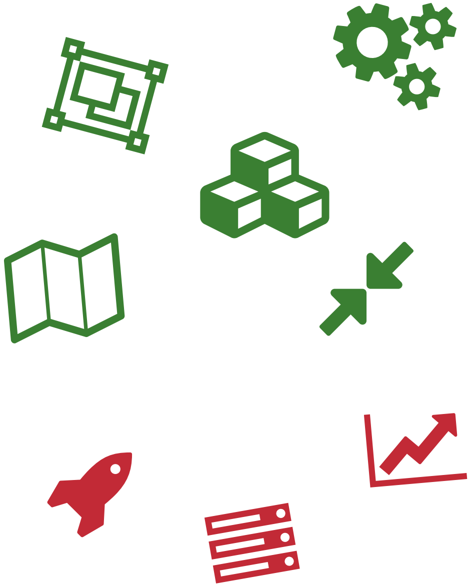
Overview

What I will talk about:

- Compositionality
- Modularity
- Genericity
- Design
- Semantics

What I wont talk about:

- Efficiency
- Optimization
- Computation Complexity

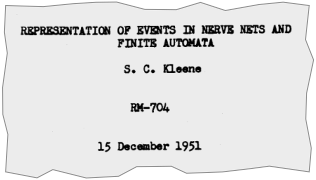


Background

- Goncharov, "Shades of Iteration: From Elgot to Kleene", WADT 2022
- Goncharov and Uustalu, "A Unifying Categorical View of Nondeterministic Iteration and Tests", CONCUR 2024

Kleene Iteration in Kleene Algebra

Regular Events

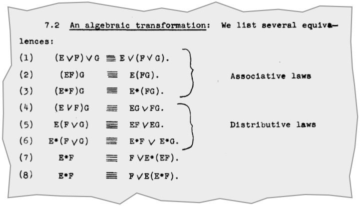


REPRESENTATION OF EVENTS IN NERVE NETS AND FINITE AUTOMATA

S. C. Kleene

RM-704

15 December 1951

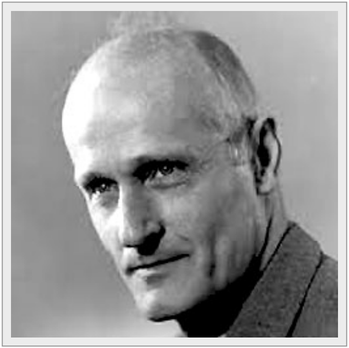


7.2 An algebraic transformation: We list several equivalences:

(1) (E ∨ F) ∨ G ≡ E ∨ (F ∨ G).
(2) (EF)G ≡ E(FG).
(3) (E*F)G ≡ E*(FG).
(4) (E ∨ F)G ≡ EG ∨ FG.
(5) E(F ∨ G) ≡ EF ∨ EG.
(6) E*(F ∨ G) ≡ E*F ∨ E*G.
(7) E*F ≡ F ∨ E*(EF).
(8) E*F ≡ F ∨ E(E*F).

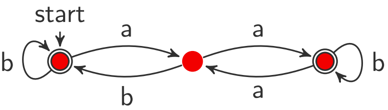
(1)–(3): Associative laws

(4)–(6): Distributive laws



- Kleene star $e \mapsto e^*$
- Kleene theorem
  - Syntax for finite state machines
  - Algebraic equational reasoning

E.g. $(b + a(ab^*a)b)^*(1 + aa)$



Language Interpretation

Regular expressions over Σ:

$$e, e_1, e_2 ::= (a \in \Sigma) \mid 0 \mid 1 \mid e_1 + e_2 \mid e_1 ; e_2 \mid e^*$$

- Language interpretation:

$$[\![0]\!] = \{\ \} \qquad [\![e_1 ; e_2]\!] = \{xy \mid x \in [\![e_1]\!], y \in [\![e_2]\!]\}$$

$$[\![1]\!] = \{\epsilon\} \qquad [\![e_1 + e_2]\!] = [\![e_1]\!] \cup [\![e_2]\!]$$

$$[\![e^*]\!] = \{\epsilon\} \cup [\![e]\!] \cup [\![e ; e]\!] \cup \ldots$$

- Language $L \subseteq \Sigma^\star$ is regular iff $L = [\![e]\!]$ for some regular expression e with $[\![a]\!] = a$ for $a \in \Sigma$

? Other interpretations? Yes, e.g. relational one!

? Complete reasoning system for regular expressions

Axioms of Kleene Algebra

Kleene algebra is a structure $(S, 0, 1, +, ;, (-)^*)$, where $(S, 0, 1, +, ;)$ is an idempotent semiring:

- $(S, 0, +)$ and $(S, 1, ;)$ are monoids
- $(S, 0, +)$ is commutative $(x + y = y + x)$ and idempotent $(x + x = x)$
- distributive laws:

$$x; (y + z) = x; y + x; z \qquad x; 0 = 0$$
$$(x + y); z = x; z + y; z \qquad 0; x = 0$$

(thus, S is partially ordered: $x \leqslant y$ iff $x + y = y$)

... plus Kleene iteration satisfying $x^* = 1 + x; x^*$, and

$$\frac{x; y + z \leqslant y}{x^*; z \leqslant y} \qquad \frac{x + z; y \leqslant z}{x; y^* \leqslant z}$$

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

Key (Design) Features

- Complete both over language model and over relational model
- Algebraic, i.e. closed under substitution, unlike Salomaa's rule*

$$\frac{y = z + xy \qquad x \ \text{ guarded}}{y = x^* z}$$

- All fixpoints are least (pre-)fixpoints
  - in Salomaa's system: particular fixpoints are unique fixpoints
- Induction rules

$$\frac{x; y + z \leqslant y}{x^*; z \leqslant y} \qquad\qquad \frac{x + z; y \leqslant z}{x; y^* \leqslant z}$$

encompass infinitely many identities, critical for completeness

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Tests for Control

- Intuition: 0 is a deadlock, 1 is a neutral program, ; is sequential composition, $+$ is non-deterministic choice
- Kleene algebra with tests (KAT) adds control via tests:
  - Kleene sub-algebra B
  - B is Boolean algebra under $(0, 1, ;, +)$
- This enables encodings:
  - Branching $(\text{if } b \text{ then } p \text{ else } q)$ as $b; p + \overline{b}; q$
  - Looping $(\text{while } b \text{ do } p)$ as $(b; p)^*; \overline{b}$
  - Hoare triples $\{a\}\, p\, \{b\}$ as $a; p; b = a; p$

Example:

$$\text{while } b \text{ do } p = \text{if } b \text{ then } p \text{ else } (\text{while } b \text{ do } p)$$

Kleene Algebra Today

- 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 ⇒ modal Kleene algebra (Struth et al.)
  - stateful ⇒ KAT + B! (Grathwohl, Kozen, Mamouras)
  - concurrent ⇒ concurrent Kleene algebra (Hoare et al.)
  - nominal ⇒ nominal Kleene algebra (Kozen et al.)
  - differential equations ⇒ differential dynamic logic (Platzer et al.)
  - network primitives ⇒ NetKAT (Foster et al.)
  - etc., etc., etc.
- decidability and completeness (most famously w.r.t. language interpretation and relational interpretation)

Beyond Kleene Algebra's Iteration

Scenario I: Exceptions

- Assumming that programs may raise exceptions: $\text{raise}\, e_i =$ "raise exception e_i",

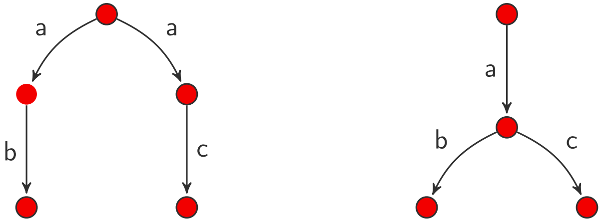
$$\text{raise}\, e_1 = \text{raise}\, e_1 ; 0 = 0 = \text{raise}\, e_2 ; 0 = \text{raise}\, e_2$$

- So, we cannot have more than one exception
  - ... unless we discard the law

$$p ; 0 = 0$$

Scenario II: Branching Time

Processes



are famously non-bisimular, failing Kleene algebra law

$$p; (q + r) = p; q + p; r$$

Scenario III: Divergence

- Identity

$$(p + 1)^* = p^*$$

  is provable in Kleene algebra, because p^* is a least fixpoint

- Alternatively:

$$1^* = 1$$

- Hence deadlock = divergence

? How to undo this?

! $1^* = 1$ is not a Kleene algebra axiom

What is generic core of Kleene iteration?

- Core reasoning principles
- Robustness under adding features (e.g. exceptions)
- Generic completeness argument
- Compatibility with classical program semantics
  - $\Rightarrow$ Soundness of while-loop encoding

Categorifying Iteration

From Algebras to Categories

- Categories $\approx$ many-sorted monoids:

$$1_A : A \to A \quad \text{(unit)} \qquad \frac{p : A \to B \qquad q : B \to C}{p; q : A \to C} \text{ (multiplication)}$$

  - Objects $A, B, \ldots$ – sorts, Morphisms $p : A \to B$ – programs
  - **Fact:** monoid = single-object category
- Kleene-Kozen categories – additionaly

$$0_{A,B} : A \to B \qquad \frac{p : A \to B \qquad q : A \to B}{p + q : A \to B} \qquad \frac{p : A \to A}{p^* : A \to A}$$

subject to Kleene algebra laws

  - **Fact:** Kleene algebra = single-object Kleene-Kozen category
  - **Example:** Category of relations = relational interpretation
- Tests = particular morphisms $b : A \to A$

Monads

Monad T ($\simeq$ Kleisli tripple)

- assigns object TA to every object A
- defines unit morphisms $\eta_A : A \to TA$
- lifts every $f : A \to TB$ to $f^\star : TA \to TB$

(monad laws omitted)

We thus can compose Kleisli morphisms $\rightsquigarrow$ Kleisli category:

$$\frac{p : A \to TB \qquad q : B \to TC}{p; q^\star : A \to TC}$$

Example: $T = \mathcal{P}$, Kleisli category $\simeq$ category of relations

Definition: Kleene monads are those, whose Kleisli category is Kleene-Kozen

Kleene Monads

Monads help us to make "robustness" idea formal via monad transformers

- Kleene monads are closed (robust) under writer transformer:

$$\mathsf{T} \mapsto \mathsf{T}(A^\star \times -)$$

- Kleene monads are **not** closed under exception transformer:

$$\mathsf{T} \mapsto \mathsf{T}(- + \mathsf{E})$$

- ... also **not** closed under coalgebraic resumption transformer:

$$\mathsf{T} \mapsto \nu\gamma.\,\mathsf{T}(- + A \times \gamma)$$

A candidate for may-diverge Kleene algebra: noting that $\mathcal{P}X \cong \{0,1\}^X$, take $\mathsf{T}X = \{0,1,\infty\}^X$

Then consider $\mathsf{Hom}(1, \mathsf{T}1) \rightsquigarrow 1^* \neq 1$ because $1 \neq \infty$

Coproducts and Elgot Iteration

- Coproducts $A \oplus B$ can be thought of as disjoint unions $A \uplus B$
- Elgot iteration:

$$\frac{p\colon A \to B \oplus A}{p^\dagger\colon A \to B}$$

  Intuitively: keep running p until reached a result in B
- $(-)^\dagger$ is subject to rich and elaborated equational theory of iteration*
  - ☺ Very general
  - ☺ Stable under adding features
  - ☺ Does not hinge on non-determinism
  - ☹ Hinges on coproducts
  - ☹ Quasi-equational axiomatizations little explored

*S. Bloom, Z. Ésik, Iteration Theories, 1993

Uniformity

Uniformity rule

$$
\begin{array}{ccc}
A & \xrightarrow{\ f\ } & B \oplus A \\
{\scriptstyle h}\downarrow & & \downarrow{\scriptstyle 1 \oplus h} \\
C & \xrightarrow{\ g\ } & B \oplus C
\end{array}
\quad \Rightarrow \quad
\begin{array}{ccc}
A & \xrightarrow{\ f^{\dagger}\ } & B \\
{\scriptstyle h}\downarrow & \nearrow_{g^{\dagger}} & \\
C & &
\end{array}
$$

for "well-behaved" h

Bloom and Esik's iteration $=$ $\underbrace{\text{Conway identities}}_{\text{finitely many}}$ $+$ $\underbrace{\text{commutative identities}}_{\text{infinitely many}}$

$$\underbrace{\text{Commutative identities}}_{\text{hard}} \subseteq \underbrace{\text{Uniformity rule}}_{\text{simple, standard}}$$

Uniform Elgot iteration is essentially just as robust and general

Reaxiomatizing Kleene Algebra

Alternative axiomatization: idempotent semirings, plus

$$p^* = 1 + p; p^* \qquad (p + q)^* = p^*; (q; p^*)^*$$

$$1^* = 1 \qquad \frac{u; p = q; u}{u; p^* = q^*; u}$$

- This is true for Kleene-Kozen categories, hence for Kleene algebra
- Removing $1^* = 1$ yields may-diverge Kleene algebras, $(-)^*$ is no longer least fixpoint
- Uniformity

$$\frac{u; p = q; u}{u; p^* = q^*; u}$$

is postulated for all u (!)

Restricting Uniformity

Like originally, u in

$$\frac{u; p = q; u}{u; p^* = q^*; u}$$

must generally be "well-behaved"

Restricting Uniformity

$$\frac{\mathsf{raise}\ \ e = \mathsf{raise}\ \ e; 1 = 1; \mathsf{raise}\ \ e = \mathsf{raise}\ \ e}{\boxed{\mathsf{raise}\ \ e} = \mathsf{raise}\ \ e; 1^* = \boxed{1^*; \mathsf{raise}\ \ e}}$$

Restricting Uniformity

$$\frac{\mathsf{raise}\ \ e = \mathsf{raise}\ \ e; 1 = 1; \mathsf{raise}\ \ e = \mathsf{raise}\ \ e}{\boxed{\mathsf{raise}\ \ e} = \mathsf{raise}\ \ e; 1* = \boxed{1*; \mathsf{raise}\ \ e}}$$

Restricting Uniformity

Like originally, u in

$$\frac{u; p = q; u}{u; p^* = q^*; u}$$

must generally be "well-behaved"

$\Rightarrow$ Restrict to linear u:

$$u; 0 = 0 \qquad u; (p + q) = u; p + u; q$$

KiCT

Kleene-iteration category with tests (KiCT)

- Category with coproducts and nondeterminism
- Selected class of tests
- Selected class of linear tame morphisms
- Kleene iteration
- Laws:

$$0;p = 0 \qquad (p+q);r = p;r + q;r$$

$$p^* = 1 + p;p^* \qquad (p+q)^* = p^*;(q;p^*)^*$$

$$\frac{u;p^* = q^*;u}{u;p = q;u}$$

with tame u

Key Results

- KiCT + $(1^* = 1)$ with all morphisms tame = Kleene-Kozen with tests and coproducts
- KiCT with expressive tests = tame-uniform Conway iteration + non-determinism
- Free KiCT = non-deterministic rational trees w.r.t. may-diverge nondeterminism

What is generic core of Kleene iteration?

KiCT:

- ✓ Core reasoning principles
- ✓ Robustness under adding features
- ✓ Generic completeness argument
- ✓ Compatibility with classical program semantics

What is generic core of Kleene iteration?

KiCT:

- ✓ Core reasoning principles
- ✓ Robustness under adding features
- ✓ Generic completeness argument
- ✓ Compatibility with classical program semantics

But what is KiCT without coproducts?

Coproducts and Non-Local Flow

What coproducts mean algebraically:

$$\mathsf{inl}; [p, q] = p \quad \mathsf{inr}; [p, q] = q \quad [\mathsf{inl}, \mathsf{inr}] = 1 \quad [p, q]; r = [p; r, q; r]$$

This creates "non-local flow", i.e. via its type $A_1 \oplus \ldots \oplus A_n$ program can switch between tracks

This can be used to derive new identities, e.g.

$$p^* = (p; (1 + p))^*$$

Alternatively to coproducts we could use names, e.g.

$$\mu X.\, (a; \mu Y.\, (b; X + 1) + 1) \quad \text{for} \quad \mathsf{inl}; [a; \mathsf{inr}, b; \mathsf{inl}]^*$$

etc.

Milner's Conundrum

- Milner* realized that "regular behaviours" are properly more general than "$*$-behaviours"
- Simplest example

$$\begin{cases} X = 1 + a; Y \\ Y = 1 + b; X \end{cases}$$

We can pass to $X = 1 + a; (1 + b; X)$, but not to $X = (ab)^*(1 + a)$

- This descrepancy $\approx$ failure of Kleene theorem
- Milner's solution is equivalent to using coproducts in the language
- He also proposed a modification of Salomaa's system for $*$-behaviours – proven complete only recently (Grabmayer)

*R. Milner, A complete inference system for a class of regular behaviours, 1984

Conclusions

- KiCTs reframe Kleene algebra principles in categorical setting and succeed with various yardsticks
- KiCTs without coproducts would be a hypothetical most basic notions of Kleene iteration
- **Open Problem:** Can it ever be found?

Appendix

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

$$1 + a(ba)^*b = 1 + a(1 + (ba)(ba)^*)b$$

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

$$\begin{aligned} 1 + a(ba)^*b &= 1 + a(1 + (ba)(ba)^*)b \\ &= 1 + a1b + a(ba)(ba)^*b \end{aligned}$$

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

$$\begin{aligned} 1 + a(ba)^*b &= 1 + a(1 + (ba)(ba)^*)b \\ &= 1 + a1b + a(ba)(ba)^*b \\ &= 1 + ab + (ab)a(ba)^*b \end{aligned}$$

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

$$\begin{aligned} 1 + a(ba)^*b &= 1 + a(1 + (ba)(ba)^*)b \\ &= 1 + a1b + a(ba)(ba)^*b \\ &= 1 + ab + (ab)a(ba)^*b \\ &= 1 + (ab)(a(ba)^*b) \end{aligned}$$

---

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

$$\begin{aligned}\boxed{1 + a(ba)^*b} &= 1 + a(1 + (ba)(ba)^*)b \\ &= 1 + a1b + a(ba)(ba)^*b \\ &= 1 + ab + (ab)a(ba)^*b \\ &= 1 + (ab)(\boxed{1 + a(ba)^*b})\end{aligned}$$

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = 1 + a(ba)^*b$$

is true, because $1 + a(ba)^*b$ is a fixpoint of the map that defines $(ab)^*$

$$\begin{aligned}\boxed{1 + a(ba)^*b} &= 1 + a(1 + (ba)(ba)^*)b \\ &= 1 + a1b + a(ba)(ba)^*b \\ &= 1 + ab + (ab)a(ba)^*b \\ &= 1 + (ab)(\boxed{1 + a(ba)^*b})\end{aligned}$$

- This only works because $x \mapsto 1 + abx$ is guarded
- $x \mapsto 1 + (a + 1)x$ is un-guarded and has infinitely many fixpoints

This reasoning is complete for guarded iteration*

? What about general (Kleene) iteration?

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

Salomaa's Complete Axiomatization

- e is guarded if
  - e is a letter
  - $e = 0$
  - $e = e_1 e_2$ with e_1 or e_2 guarded
  - $e = e_1 + e_2$ with e_1 and e_2 guarded
- Salomaa originally defined dual empty word property (ewp): e has epw iff it is not guarded
- ... and, proposed complete axiomatization* w.r.t. language model:
  - A finite number of sound identities
  - plus rule:

$$\frac{v = e + uv \qquad u \text{ guarded}}{v = u^* e}$$

*A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events, 1966

No Finite Equational Axiomatization

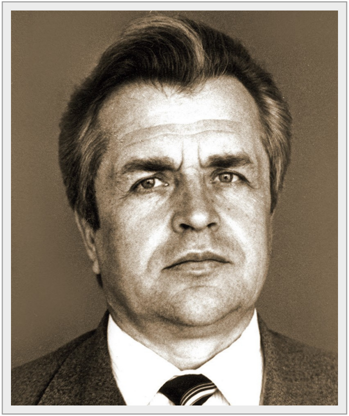
Redko* noticed that

- All identities (power identities)

$$e^* = (e^k)^*(1 + e + \ldots + e^{k-1})$$

  are sound
- Any finite set of sound equations entails only finitely many of them
- Hence, no finite axiomatizability

  (even on one-letter alphabet)



So,

? How to choose infinite set of non-obvious axioms of iteration?

? How would we know that this choice is correct?

---

*V. N. Redko, On defining relations for the algebra of regular events, 1964

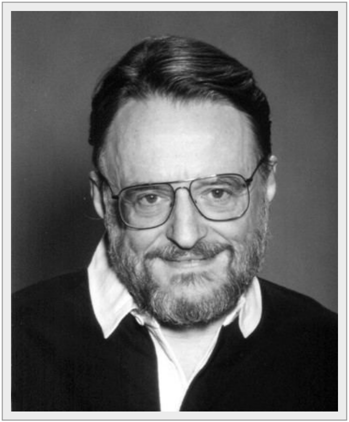
Conway's Monograph

Conway* came up with various insights:

- Power identities do not suffice, e.g. they do not imply

$$(e+u)^* = \Big((e+u)(u+(eu^*)^{n-2}e)\Big)^* \Big(1+(e+u)\sum_{i=0}^{n-2}(eu^*)^i\Big)$$

- Made several conjectures on potential complete axiomatization
- Observed that algebraic laws of regular expressions transfer to matrices of regular expressions



⇒ Bridge between algebra and automata (represented by matrices)

---

*J. H. Conway, Regular Algebra and Finite Machines, 1971

Matrices of Regular Expressions

- $(n \times n)$-matrices of regular expressions support same operations.
  For $n = 2$:

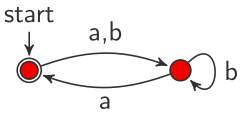
"1" is $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = \begin{bmatrix} a + a' & b + b' \\ c + c' & d + d' \end{bmatrix}$

"0" is $O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ $\qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = \begin{bmatrix} aa' + bc' & ab' + bd' \\ ca' + dc' & cb' + dd' \end{bmatrix}$

- **Idea** for A^*: $I + A + A^2 + \ldots$
  Key insight: there is closed form for A^* as matrix of regular expressions

- **Intuition**: in $\begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} = A^*$, e_{ij} represents language of 2-state automaton where i – initial, j – final

Automata and Matrices



$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^{\top}\begin{bmatrix}0 & a+b\\ a & b\end{bmatrix}^{*}\begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^{\top}\begin{bmatrix}((a+b)b^*a)^* & ((a+b)b^*a)^*\\ (b^*a(a+b))^*a & b^*(a(a+b))^*\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

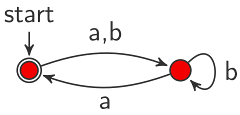
$$((a+b)b^*a)^*$$

Automata and Matrices

- Automata are triples

$$A \in \{0,1\}^n,\ B \in \mathcal{E}^{n\times n},\ C \in \{0,1\}^n$$

$\mathcal{E}$ – certain class of regular expressions



$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^\top \begin{bmatrix}0 & a+b\\ a & b\end{bmatrix}^* \begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^\top \begin{bmatrix}((a+b)b^*a)^* & ((a+b)b^*a)^*\\ (b^*a(a+b))^*a & b^*(a(a+b))^*\end{bmatrix} \begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$((a+b)b^*a)^*$$

Automata and Matrices

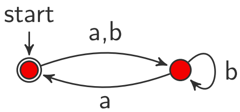
- Automata are triples

$$A \in \{0,1\}^n,\ B \in \mathcal{E}^{n\times n},\ C \in \{0,1\}^n$$

$\mathcal{E}$ – certain class of regular expressions

- Accepted language:

$$[\![A^\top B^* C]\!]$$



$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^\top \begin{bmatrix}0 & a+b\\ a & b\end{bmatrix}^* \begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^\top \begin{bmatrix}((a+b)b^*a)^* & ((a+b)b^*a)^*\\ (b^*a(a+b))^*a & b^*(a(a+b))^*\end{bmatrix} \begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$((a+b)b^*a)^*$$

Automata and Matrices

- Automata are triples

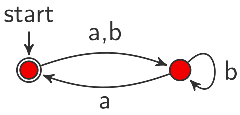
$$A \in \{0,1\}^n, \ B \in \mathcal{E}^{n \times n}, \ C \in \{0,1\}^n$$

$\mathcal{E}$ – certain class of regular expressions

- Accepted language:

$$[\![A^\top B^* C]\!]$$

- Kleene theorem: this is equivalence between automata and expressions up to language equality



$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^\top \begin{bmatrix}0 & a+b\\ a & b\end{bmatrix}^* \begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$\begin{bmatrix}1\\0\end{bmatrix}^\top \begin{bmatrix}((a+b)b^*a)^* & ((a+b)b^*a)^*\\ (b^*a(a+b))^*a & b^*(a(a+b))^*\end{bmatrix} \begin{bmatrix}1\\0\end{bmatrix}$$

$$\Updownarrow$$

$$((a+b)b^*a)^*$$

Control in Category

- Call morphisms of the form $\mathrm{d}\colon A \to A \oplus A$ decisions
  - In particular: ff – left injection, tt – right injection
- We then can express if-then-else:

$$\frac{\mathrm{d}\colon A \to A \oplus A \qquad \mathrm{p}\colon A \to \mathrm{B} \qquad \mathrm{q}\colon A \to \mathrm{B}}{\underline{\text{if}}\ \mathrm{d}\ \underline{\text{then}}\ \mathrm{p}\ \underline{\text{else}}\ \mathrm{q}\colon A \to \mathrm{B}}$$

  - In particular: $\sim\mathrm{d} = \underline{\text{if}}\ \mathrm{d}\ \underline{\text{then}}\ \mathrm{ff}\ \underline{\text{else}}\ \mathrm{tt}$, $(\mathrm{d} \parallel e) = \underline{\text{if}}\ \mathrm{d}\ \underline{\text{then}}\ \mathrm{tt}\ \underline{\text{else}}\ e$
- Various expected laws are entailed, but some are not, e.g.

$$\mathrm{d} \parallel \mathrm{tt} \neq \mathrm{tt}$$

Uniform Conway While-Operator

Theorem*: if the class of decisions is large enough, uniform Conway iteration is equivalent to while-loops

Axioms:

$$\underline{\mathsf{while}}\ d\ \underline{\mathsf{do}}\ p = \underline{\mathsf{if}}\ d\ \underline{\mathsf{then}}\ p; (\underline{\mathsf{while}}\ d\ \underline{\mathsf{do}}\ p)\ \underline{\mathsf{else}}\ 1$$

$$\underline{\mathsf{while}}\ (d \parallel e)\ \underline{\mathsf{do}}\ p = (\underline{\mathsf{while}}\ d\ \underline{\mathsf{do}}\ p); \underline{\mathsf{while}}\ e\ \underline{\mathsf{do}}\ (p; \underline{\mathsf{while}}\ d\ \underline{\mathsf{do}}\ p)$$

$$\underline{\mathsf{while}}\ (d\ \&\&\ (e \parallel tt))\ \underline{\mathsf{do}}\ p = \underline{\mathsf{while}}\ d\ \underline{\mathsf{do}}\ (\underline{\mathsf{if}}\ e\ \underline{\mathsf{then}}\ p\ \underline{\mathsf{else}}\ p)$$

Uniformity Rule:

$$\frac{u; \underline{\mathsf{if}}\ d\ \underline{\mathsf{then}}\ p; tt\ \underline{\mathsf{else}}\ \ ff = \underline{\mathsf{if}}\ e\ \underline{\mathsf{then}}\ q; u; tt\ \underline{\mathsf{else}}\ v; ff}{u; \underline{\mathsf{while}}\ d\ \underline{\mathsf{do}}\ p = (\underline{\mathsf{while}}\ e\ \underline{\mathsf{do}}\ q); v}$$

where u, v come from a selected class of programs

*S. Goncharov, Shades of Iteration: From Elgot to Kleene, 2023

Tests and Decisions

- In presence of non-determinism, decisisons $\mathrm{d}\colon A \to A \oplus A$ decompose:

$$\mathrm{d} = \mathrm{b}; \mathrm{tt} + \bar{\mathrm{b}}; \mathrm{ff} \qquad\qquad (\mathrm{b}, \bar{\mathrm{b}}\colon A \to A)$$

- Test-based 'if' and 'while':

Axioms:

$$\mathsf{while}\ \mathrm{b}\ \mathsf{do}\ \mathrm{p} = \mathsf{if}\ \mathrm{b}\ \mathsf{then}\ \mathrm{p}; (\mathsf{while}\ \mathrm{b}\ \mathsf{do}\ \mathrm{p})\ \mathsf{else}\ 1$$

$$\mathsf{while}\ (\mathrm{b} \vee \mathrm{c})\ \mathsf{do}\ \mathrm{p} = (\mathsf{while}\ \mathrm{b}\ \mathsf{do}\ \mathrm{p}); \mathsf{while}\ \mathrm{c}\ \mathsf{do}\ (\mathrm{p}; \mathsf{while}\ \mathrm{b}\ \mathsf{do}\ \mathrm{p})$$

Uniformity:

$$\frac{u; \mathrm{b}; \mathrm{p} = \mathrm{c}; \mathrm{q}; u \qquad u; \bar{\mathrm{b}} = \bar{\mathrm{c}}; v}{u; \mathsf{while}\ \mathrm{b}\ \mathsf{do}\ \mathrm{p} = (\mathsf{while}\ \mathrm{c}\ \mathsf{do}\ \mathrm{q}); v}$$