Pushing the Limits of Kleene Algebra

Sergey Goncharov

FAU Erlangen-Nürnberg

RAMICS-AiML 2024, 19-22 August 2024

Overview

What I will talk about:

- Compositionality
- Modularity
- Genericity
- Design
- Semantics

What I wont talk about:

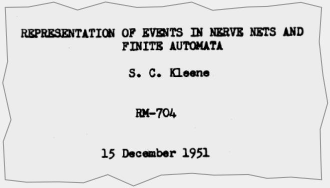
- Efficiency
- Optimization
- Computation Complexity

Background

- Goncharov, “Shades of Iteration: From Elgot to Kleene”, 2023, WADT 2022
- Goncharov and Uustalu, “A Unifying Categorical View of Nondeterministic Iteration and Tests”, 2024, CONCUR 2024

Historical Overture

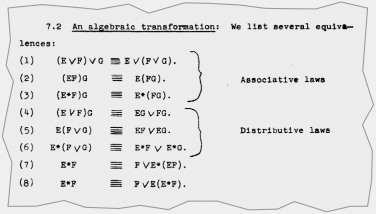
Regular Events


REPRESENTATION OF EVENTS IN NERVE NETS AND FINITE AUTOMATA

S. C. Kleene

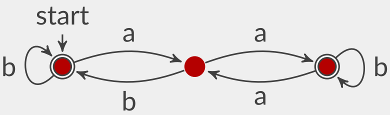
RM-704

15 December 1951


7.2 An algebraic transformation: We list several equivalences:

(1)	$(E \vee F) \vee G$	$\equiv$	$E \vee (F \vee G)$.	Associative laws
(2)	$(EF)G$	$\equiv$	$E(FG)$.	
(3)	$(E*F)G$	$\equiv$	$E*(FG)$.	
(4)	$(E \vee F)G$	$\equiv$	$EG \vee FG$.	Distributive laws
(5)	$E(F \vee G)$	$\equiv$	$EF \vee EG$.	
(6)	$E*(F \vee G)$	$\equiv$	$E*F \vee E*G$.	
(7)	$E*F$	$\equiv$	$F \vee E*(EF)$.	
(8)	$E*F$	$\equiv$	$F \vee E(E*F)$.	

$$\boxed{(b + a(ab^*a)b)^*(1 + aa)}$$



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

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:

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

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

$$[\![e^*]\!] = \{\epsilon\} \cup [\![e]\!] \cup [\![ee]\!] \cup \dots$$

- 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

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = \mathbf{1} + a(ba)^* b$$

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

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = \mathbf{1} + a(ba)^*b$$

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

$$\mathbf{1} + a(ba)^*b = \mathbf{1} + a(\mathbf{1} + (ba)(ba)^*)b$$

Equivalence of Expressions

Example proof "by coinduction":

$$(ab)^* = \mathbf{1} + a(ba)^*b$$

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

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

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 $(ab)^*$

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

Example proof "by coinduction":

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

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

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

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 $(ab)^*$

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

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

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:

standalone/salomaa-photo.

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

*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)

standalone/redko-photo.jp

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

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

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

standalone/conway-photo.j

$\Rightarrow$ Bridge between algebra and automata (represented by matrices)

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

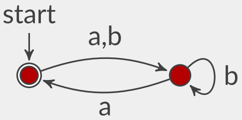
Matrices of Regular Expressions

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

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

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

- **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



$$\Updownarrow$$

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

$$\Updownarrow$$

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

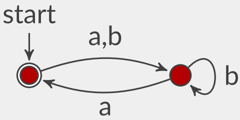
$$\Updownarrow$$

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

- Automata 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

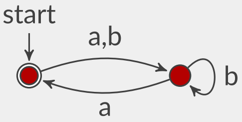
- Automata are triples

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

$\mathcal{E}$ – 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

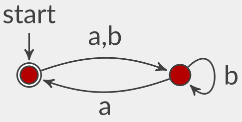
$$A \in \{0,1\}^n,\ B \in \mathcal{E}^{n\times n},\ C \in \{0,1\}^n$$

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

- Accepted language:

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

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



$$\Updownarrow$$

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

$$\Updownarrow$$

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

$$\Updownarrow$$

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

Kleene Algebra

- Kozen defined Kleene algebra:

  1. Idempotent semiring
  2. e^*u – least pre-fixpoint of $u + e(-)$:

     $$u + ee^*u = e^*u \qquad \frac{u + ew \leqslant w}{e^*u \leqslant w}$$

     using: $x \leqslant y$ iff $x + y = y$
  3. ue^* – least pre-fixpoint of $u + (-)e$

- **Completeness:** given $[\![e_1]\!] = [\![e_2]\!]$,
  1. $e_1 \rightsquigarrow A_1^\top B_1^* C_1, \quad e_2 \rightsquigarrow A_2^\top B_2^* C_2$
  2. eliminate ϵ-transitions
  3. determinize
  4. minimize
  5. show that all ensuing transitions $A^\top B^* C = \hat{A}^\top \hat{B}^* \hat{C}$ are provable

- **Corollary:** Language interpretation = free Kleene algebra

standalone/kozen-photo.jp

Key Design Features

- Not tailored to language model – complete also over relational model
- Algebraic, i.e. closed under substitution, in contrast to Salomaa's rule

$$\frac{w = u + ew \quad e \text{ guarded}}{w = e^* u}$$

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

$$\frac{u + ew \leqslant w}{e^* u \leqslant w} \qquad\qquad \frac{u + we \leqslant w}{ue^* \leqslant w}$$

encompass infinitely many identities, critical for completeness

Tests for Control

- Another reading: Algebra elements = programs
  - 0 – divergence and/or deadlock, 1 – neutral program, etc.
- 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 $\Rightarrow$ 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.)
  - differential equations $\Rightarrow$ differential dynamic logic (Platzer et al.)
  - network primitives $\Rightarrow$ NetKAT (Foster et al.)
  - etc., etc., etc.
- decidability and completeness (most famously w.r.t. language interpretation and relational interpretation)

Pushing Limits

Scenario I: Exceptions

- Assumming programs raise exceptions: $\text{raise}\, e_i$ = “raise exception e_i”,

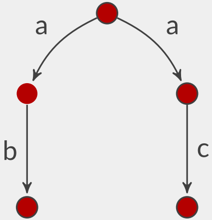
$$\text{raise}\, e_1 = \text{raise}\, e_1; 0 = 0 = \text{raise}\, e_2; 0 = \text{raise}\, e_2$$

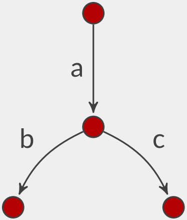
- So, we cannot have more than one exception
  - ... unless we discard the law

$$p; 0 = 0$$

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

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

What is generic core of Kleene iteration?

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

Categorifying Iteration

From Algebras to Categories

- Categories $\approx$ many-sorted monoids:

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

  - 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 $b : A \to A$

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

standalone/esik-photo.jpg

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

Coproducts and Elgot Iteration

- Coproducts $A \oplus B$ can be thought of as disjoint un

There is only one theory of iteration

- Elgot iteration:

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

  - 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

standalone/esik-photo.jpg

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

Program Features

- Given Elgot iteration operator, fix carrier of exceptions E
- Exception-raising morphisms $A \to B \oplus E$ themeselves form a category
- Elgot iteration and its laws carry over
  - This fails for Kleene-Kozen categories
- Elgot iteration's laws are thus stable under exception monad transformer
- Similarly: state, reading, writing, adjoining process algebra actions

Uniform Conway Iteration

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

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

? Can we formulate uniform Conway iteration via familiar while-loops

Control in Category

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

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

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

$$d \mathbin{||} \mathrm{tt} \neq \mathrm{tt}$$

Uniform Conway While-Operator

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

Axioms:

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

$$\underline{\text{while}}\ (d \;\|\; e)\ \ \underline{\text{do}}\ p = (\underline{\text{while}}\ d\ \ \underline{\text{do}}\ p); \underline{\text{while}}\ e\ \ \underline{\text{do}}\ (p; \underline{\text{while}}\ d\ \ \underline{\text{do}}\ p)$$

$$\underline{\text{while}}\ (d\ \&\&\ (e \;\|\; \mathrm{tt}))\ \ \underline{\text{do}}\ p = \underline{\text{while}}\ d\ \ \underline{\text{do}}\ (\underline{\text{if}}\, e\, \underline{\text{then}}\, p\, \underline{\text{else}}\ p)$$

Uniformity Rule:

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

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 $d\colon A \to A \oplus A$ decompose:

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

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

Axioms:

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

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

Uniformity:

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

Reaxiomatizing Kleene Algebra

Alternative axiomatization: idempotent semiring, and

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

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

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

Restricting Uniformity

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

Restricting Uniformity

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

Restricting Uniformity

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

❗ Need not hold in may-diverge Kleene algebras

Restricting Uniformity

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

❗ Need not hold in may-diverge Kleene algebras
$\Rightarrow$ Restrict to linear u:

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

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

Properties

- 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

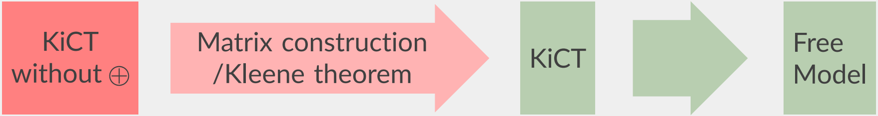
What is generic core of Kleene iteration?

KiCT:

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

But what is KiCT without coproducts?

Hypothetical Route



- If everything is tame (Kleene algebra), this is essentially what happens
- What if nothing is tame (Process algebra)?

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 matrix construction/Kleene theorem

standalone/milner-photo.j

- 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?