Concurrent Kleene algebra with tests

Peter Jipsen

School of Computational Sciences and
Center of Excellence in Computation, Algebra and Topology (CECAT)
Chapman University, Orange, California

April 28, 2014

Outline

- Short review of **Kleene Algebras** (KA), **KA with Tests** (KAT) and **Concurrent KA** (CKA)
- Generalize to **Concurrent KAT** (CKAT)
- Automata for **guarded series-parallel strings**
- **Trace semantics** for CKAT
- **Concurrent relation algebras** with transitive closure

Introduction

Kleene algebras with tests (KAT) are defined by Kozen and Smith in 1997 as **Kleene algebras** with a subalgebra of **Boolean tests**, with semantics based on **guarded strings**

Concurrent Kleene algebras (CKA) are introduced by Hoare, Möller, Struth and Wehrman in 2009 as **idempotent bisemirings** that satisfy a **concurrency inequation** and have a **Kleene-star** for both **sequential** and **concurrent composition**

Concurrent Kleene algebras with tests **(CKAT)** combine these concepts

Guarded strings are generalized to ***guarded series-parallel strings*** (gsp-strings)

Sets of **gsp-strings** provide a concrete **language model for CKAT**

Guarded automata of Kozen [2003] combined with

branching automata of Lodaya and Weil [2000]

$\Longrightarrow$ a model for computing in parallel on gsp-strings

$\Longrightarrow$ trace semantics for simple concurrent computations

Motivation

Relation algebras and **Kleene algebras with tests** can model **specifications** and **programs**

Automata and **coalgebras** can model **state based systems** and **object-oriented programs**

These paradigms are well suited for **single threaded computations**

Multi-core architectures and **cluster-computing** are now widely available

The recent development of **concurrent Kleene algebra** (**CKA**) builds on a computational model (KA) that is elegant and has numerous applications

Useful to explore which aspects of **Kleene algebras with tests** can be lifted easily to a **concurrent** setting

Preserve the **simplicity** of **regular languages** and **(guarded) strings**

For the nonguarded case many interesting results have been obtained by Lodaya and Weil [2000] using **labeled posets** (or **pomsets**) of Pratt [1986] and Gisher [1988], but restricted to the class of **series-parallel pomsets** called **sp-posets**

Want to extend **guarded strings** to handle **concurrent composition** with the same approach as for sp-posets

Review of KAT

A ***Kleene algebra with tests*** (**KAT**) is an **idempotent semiring**

with a **Boolean subalgebra of tests** and

a unary **Kleene-star operation** that plays the role of **reflexive-transitive closure**

I.e., a **two-sorted algebra** of the form $\mathbf{A} = (A, A', +, 0, \cdot, 1, {}^{-}, {}^{*})$

where A' is a **subset** of A,

$(A, +, 0, \cdot, 1, {}^{*})$ is a **Kleene algebra** and

$(A', +, 0, \cdot, 1, {}^{-})$ is a **Boolean algebra**

Complementation is only defined on A'

Let Σ be a set of ***basic program symbols*** $p, q, r, p_1, p_2, \ldots$ and

T a set of ***basic test symbols*** $t, t_1, t_2, \ldots$ (assume $\Sigma \cap T = \emptyset$)

Elements of T are **Boolean generators**

write 2^T for the **set of *atomic tests***,

= **characteristic functions** on T, denoted by $\alpha, \beta, \gamma, \alpha_1, \alpha_2, \ldots$

The set of **guarded strings** over $\Sigma \cup T$ is

$$GS_{\Sigma,T} = 2^T \times \bigcup_{n<\omega} (\Sigma \times 2^T)^n$$

A typical **guarded string** is denoted by $\alpha_0 p_1 \alpha_1 p_2 \alpha_2 \ldots p_n \alpha_n$,

or by $\alpha_0 w \alpha_n$ for short, where $\alpha_i \in 2^T$ and $p_i \in \Sigma$

For **finite** T the members of $2^T \subseteq GS_{\Sigma,T}$ can be identified with the **atoms** of the **free Boolean algebra generated by** T

Concatenation of guarded strings is via the **coalesced product**:

$w\alpha \diamond \beta w' = w\alpha w'$ if $\alpha = \beta$ and **undefined** otherwise

For subsets L, M of $GS_{\Sigma,T}$ define

- $L + M = L \cup M$
- $LM = \{v \diamond w : v \in L, w \in M$ and $v \diamond w$ **is defined**$\}$
- $0 = \emptyset$
- $1 = 2^T$
- $\bar{L} = 2^T \setminus L$ **if** $L \subseteq 2^T$
- $L^* = \bigcup_{n<\omega} L^n$ where $L^0 = L$ and $L^n = LL^{n-1}$ for $n > 0$

Then $\mathcal{P}(GS_{\Sigma,T})$ **is a KAT** under these operations

Define a map G from **KAT terms** over $\Sigma \cup T$ to this concrete model by

- $G(t) = \{\alpha \in 2^T : \alpha(t) = 1\}$ for $t \in T$,
- $G(p) = \{\alpha p \beta : \alpha, \beta \in 2^T\}$ for $p \in \Sigma$,
- $G(p + q) = G(p) + G(q), \qquad G(pq) = G(p)G(q)$, $G(p^*) = G(p)^*$, for **any terms** p, q and
- $G(0) = 0,\ G(1) = 1,\ G(\bar{b}) = \overline{G(b)}$ for **any Boolean term** b.

The ***language model*** $\mathbf{G}_{\Sigma,T}$ is the **subalgebra** of $\mathcal{P}(GS_{\Sigma,T})$ **generated by** $\{G(t) : t \in T\} \cup \{G(p) : p \in \Sigma\}$

$\mathbf{G}_{\Sigma,T}$ is the free KAT and its members are the ***rational guarded languages***

Subsets of 2^T are called **Boolean tests**

Other members of $\mathbf{G}_{\Sigma,T}$ are called **programs**

A ***nondeterministic guarded automaton*** is a tuple $\mathcal{A} = (X, \delta, F)$ where

- $\delta \subseteq X \times (\Sigma \cup \mathcal{P}(2^T)) \times X$ is the **transition relation** and
- $F \subseteq X$ is the set of **final states**

$(x, t, y) \in \delta$ is a **test transition** if $t \in \mathcal{P}(2^T)$

Acceptance of a guarded string w by $\mathcal{A}$ **starting from initial state** x_0 and **ending in state** x_f is defined **recursively** by:

- If $w = \alpha \in 2^T$ then w is accepted **iff** for some $n \geq 1$ **there is a path** $x_0 t_1 x_1 t_2 \dots x_{n-1} t_n x_f$ in $\mathcal{A}$ of n **test transitions** $t_i \in \mathcal{P}(2^T)$ such that $\alpha \in t_i$ for $i = 1, \dots, n$
- If $w = \alpha p v$ then w is accepted **iff** there **exist states** x_1, x_2 such that α is accepted ending in state x_1, **there is a transition labeled** p from x_1 to x_2 (i.e., $(x_1, p, x_2) \in \delta$) and v is accepted by $\mathcal{A}$ starting from initial state x_2

Finally, w is ***accepted by*** $\mathcal{A}$ ***starting from*** x_0 if the ending state x_f is **indeed a final state**, i.e., satisfies $x_f \in F$

The ***regular guarded languages*** are sets of guarded strings that are accepted by a **finite automaton** starting from some initial state

Kleene showed that **rational languages = regular languages**; same holds for **guarded** languages

Kozen [2003] proved that the **equational theory of KAT is decidable in PSPACE**

KAT is **more versatile** that Kleene algebra

E.g. can express "**if** b **then** p **else** q" by the term $bp + \overline{b}q$ and

"**while** b **do** p" using $(bp)^*\overline{b}$

KAT also interprets **Hoare logic**

Distinguishes between simple **Boolean tests** and **complex assertions**

Adding concurrency

Now generalize these definitions to handle **concurrency**

Elements P, Q of a **concurrent Kleene algebra with tests** are programs or program fragments

They are represented by sets of "**computation paths**" (traces)

Need to add **concurrent composition** $P||Q$

In the **sequential model** the computation paths are **guarded strings**

Want to place two such sequential strings "**next to each other**"

Also need to **sequentially compose** such "**concurrent strings**" etc

View **sequential composition** as **vertical concatenation** (top to bottom) and

concurrent composition as **horizontal concatenation**

E.g., given two **guarded strings** $\alpha_0 v \alpha_m$ and $\beta_0 w \beta_n$ construct

$$\begin{array}{c||c} \alpha_0 & \beta_0 \\ v & w \\ \alpha_m & \beta_n \end{array}$$

As with **sequential composition**, this operation is **not always defined**

To be **concurrently composable**, require $\alpha_0 = \beta_0$ and $\alpha_m = \beta_n$

So we have $\alpha_0 v \alpha_m || \alpha_0 w \alpha_m$ and denote result by $\alpha_0 \{\!| v, w |\!\} \alpha_m$ or **vertically** by

$$\begin{array}{c} \alpha_0 \\ v \;\big|\; w \\ \alpha_m \end{array}$$

If α, β are **distinct atomic tests** then $\alpha || \beta$ is **undefined**

$\alpha || \alpha = \alpha$

$\alpha || \beta w \gamma$ is **undefined for all atomic tests** α, β, γ

Concurrent composition is **commutative**:

$\{\!\!| v, w |\!\!\} = \{\!\!| w, v |\!\!\}$ is a **multiset**

$||$ is **associative**, i.e., $\{\!\!| \{\!\!| u, v |\!\!\}, w |\!\!\}$ is **normalized** to $\{\!\!| u, v, w |\!\!\}$

Hence $(\alpha p \beta || \alpha q \beta) || \alpha r \beta = \alpha \{\!\!| p, q, r |\!\!\} \beta = \alpha p \beta || (\alpha q \beta || \alpha r \beta)$

Guarded series-parallel strings, or ***gsp-strings*** for short are constructed by successive **concurrent** and **sequential** compositions

Formally the **set of gsp-strings generated by** Σ, T is the smallest set $GSP_{\Sigma,T}$ that has 2^T and $2^T \times \Sigma \times 2^T$ as subsets and

is **closed** under **coalesced product** $\diamond$ and **concurrent product** $||$

E.g., if $\Sigma = \{p, q\}$ and $T = \{t\}$ then, abbreviating 2^T by $\{\alpha, \beta\}$, the following expressions are gsp-strings:

$\alpha, \quad \alpha p \alpha, \quad \alpha p \beta, \quad \alpha\{\!|p, q|\!\}\alpha, \quad \alpha\{\!|p, q|\!\}\alpha q \beta, \quad \alpha\{\!|p, \{\!|p, q|\!\}\alpha q|\!\}\beta,$

...

The **language model** over **gsp-strings** is defined as in the case of guarded strings, except that we now have an additional operation. For $L, M \in \mathcal{P}(GSP_{\Sigma,T})$ let

- $L||M = \{v||w : v \in L, w \in M$ and $v||w$ **is defined**$\}$

This makes $\mathcal{P}(GSP_{\Sigma,T})$ into a **complete bisemiring** with a **Kleene-star for sequential composition**

The map G from is extended to all terms of KAT with $||$, by defining $G(p||q) = G(p)||G(q)$

The bi-Kleene algebra $\mathbf{C}_{\Sigma,T}$ of ***rational gsp-languages*** is the subalgebra generated by $\{G(t) : t \in T\} \cup \{G(p) : p \in \Sigma\}$

Note that for $b \in \mathcal{P}(2^T)$ and for any subset p of $GSP_{\Sigma,T}$ the **concurrent composition** $b||p$ is equal to $b \cap p$

$\Longrightarrow$ concurrent and sequential composition **coincide on tests**

However, in general $||$ is **not** idempotent for sets of gsp-strings and

the identity 1 of **sequential composition** is **not** an identity of **concurrent composition**

With this language model as guide, we now define a ***concurrent Kleene algebra with tests* (CKAT)** as an algebra $\mathbf{A} = (A, A', +, 0, ||, \cdot, 1, ^*, ^-)$ where

- $(A, A', +, 0, \cdot, 1, ^*, ^-)$ is a **Kleene algebra with tests**,
- $(A, +, 0, ||)$ is a **commutative semiring with** 0 (but possibly no unit), and
- $b||c = bc$ for all $b, c \in A'$.

Iterated parallel composition (i.e., parallel star) is **not included** in CKAT

It would prevent the generalization of Kleene's theorem to gsp-languages

The language model also shows that the **concurrency inequation** $(x||y)(z||w) \leq (xz)||(yw)$ of **CKA** is **not satisfied** under the present definition of CKAT

E.g., let $x = \{\alpha p \beta\}$, $y = \{\alpha q \beta\}$, $z = \{\beta p \gamma\}$, and $w = \{\beta q \gamma\}$

Then $(x||y)(z||w) = \{\alpha \{\!| p, q |\!\} \beta \{\!| p, q |\!\} \gamma\}$

whereas $(xz)||(yw) = \{\alpha \{\!| p\beta p, q\beta q |\!\} \gamma\}$

So each expression produces a **singleton set**, but the two elements are **distinct**, hence the two expressions are **not comparable**

However one can impose the **concurrency inequation** on the generators of the **regular gsp-languages** to obtain a homomorphic image that satisfies this condition

Automata over gsp-strings

Let $\mathcal{M}(X)$ be the set of **multisets** of X with **more than one element**

A ***guarded branching automaton*** is specified by a tuple $\mathcal{A} = (X, \delta, \delta_{\text{fork}}, \delta_{\text{join}}, F)$, where

- (X, δ, F) is a **guarded automaton**,
- $\delta_{\text{fork}} \subseteq X \times \mathcal{M}(X)$ and
- $\delta_{\text{join}} \subseteq \mathcal{M}(X) \times X$

Fork transitions in δ_{fork} are denoted $(x, \{\!|x_1, x_2, \ldots, x_n|\!\})$

If the multiset has n elements they are called **forks of arity** n

Join transitions of arity n are defined by $(\{\!|x_1, x_2, \ldots, x_n|\!\}, x)$

A ***weak guarded series parallel string*** (or **wgsp-string** for short) is a gsp-string but possibly **without the first and/or last** atomic test

Acceptance of a wgsp-string w by $\mathcal{A}$ **starting from initial state** x_0 and **ending at state** x_f, is defined recursively by:

- If $w = \alpha \in 2^T$ then w is accepted **iff** for some $n \geq 1$ **there is a sequential path** $x_0 t_1 x_1 t_2 \ldots x_{n-1} t_n x_f$ in $\mathcal{A}$ (i.e., $(x_{i-1}, t_i, x_i) \in \delta$) of n test transitions $t_i \in \mathcal{P}(2^T)$ such that $\alpha \in t_i$ for $i = 1, \ldots, n$.
- If $w = p \in \Sigma$ then w is accepted **iff** there **exist a transition** labeled p from x_0 to x_f.

- If $w = \{\!| u_1, \ldots, u_m |\!\} v$ for $m > 1$ then w is accepted **iff** there **exist a fork** $(x_0, \{\!| x_1, \ldots, x_m |\!\})$ **and a join** $(\{\!| y_1, \ldots, y_m |\!\}, y_0)$ in $\mathcal{A}$ such that u_i is accepted starting from x_i and ending at y_i for all $i = 1, \ldots, m$, and furthermore βv is accepted by $\mathcal{A}$ **starting at** y_0 **and ending at** x_f.
- If $w = uv$ then w is accepted **iff** there **exist a state** x such that u is accepted **ending in state** x and v is accepted by $\mathcal{A}$ **starting from initial state** x and **ending at** x_f.

Finally, w is ***accepted by*** $\mathcal{A}$ ***starting from*** x_0 if the **ending state** $x_f \in F$

A **fork transition** corresponds to the creation of n **separate processes** that can **work concurrently** on the acceptance of the wgsp-strings $u_1, \ldots, u_n$

The matching **join transition** then corresponds to a **communication** or **merging of states** that terminates these processes and **continues in a single thread**

The sets of gsp-strings that are **accepted by a finite automaton** are called ***regular gsp-languages***

For sets of **(unguarded) strings**, the **regular languages** and the **rational languages** (i.e., those built from Kleene algebra terms) **coincide**

Loyala and Weil show that e.g. the language $\{p, p||p, p||p||p, \ldots\}$ is a **regular sp-language** , but **not a rational sp-language**

The ***width*** of an **sp-poset** or a **gsp-string** is the **maximal cardinality of an antichain** in the underlying poset

A **(g)sp-language** is said to be of ***bounded width*** if there exists $n < \omega$ such that every member of the language has **width less than** n

Intuitively this means that the language can be accepted "efficiently" by a machine that has no more than n processors

The **rational gsp-languages** are of **bounded width** since **concurrent iteration is not included** as one of the operations of **CKAT**

For **languages of bounded width** Kleene's theorem holds (Lodaya and Weil):

A sp-language is **rational** if and only if it is **regular** (i.e., accepted by a finite automaton) and has **bounded width**

Now relate the **rational sp-languages** to **rational gsp-languages**

Let $\overline{T} = \{\overline{t} : t \in T\}$ be the set of **negated basic tests**

Assume $T = \{t_1, \ldots, t_n\}$ is **finite**

Consider **atomic tests** α to be (sequential) strings of the form $b_1 b_2 \ldots b_n$ where each b_i is either the element t_i or $\overline{t}_i$

Every term p can be **transformed** into a term p' in **negation normal form** using DeMorgan laws and $\overline{\overline{b}} = b$, so that **negation only appears on t_i**

Hence the term p' is also a CKA term over the set $\Sigma \cup T \cup \overline{T}$

Let $R(p')$ be the result of evaluating p' in the set of **sp-posets** of Lodaya and Weil

Kozen and Smith show how to transform p' further to a sum $\hat{p}$ of ***externally guarded*** **terms** such that $p = p' = \hat{p}$ in KAT and $R(\hat{p}) = G(\hat{p})$

This argument **also applies to terms of CKAT** since $||$ distributes over $+$

So the completeness result of Lodaya and Weil extends as follows

Theorem 1. $\text{CKAT} \models p = q \iff G(p) = G(q)$

It follows that $\mathbf{C}_{\Sigma,T}$ is indeed the **free algebra of CKAT**

Theorem 2. A set of gsp-strings is **rational** (i.e. an element of $\mathbf{C}_{\Sigma,T}$) if and only if it is accepted by a **finite guarded branching automaton** and has **bounded width**.

A run of $\mathcal{A}$ is called ***fork-acylic*** if a matching fork-join pair **never occurs** as a matched pair **nested within itself**

$\mathcal{A}$ is ***fork-acylic*** if **all accepted runs** of $\mathcal{A}$ are fork-acyclic

Lodaya and Weil prove that if a language is accepted by a **fork-acyclic automaton** then it has **bounded width**, and their proof applies equally well to gsp-languages

Trace semantics for CKAT

Kozen and Tiuryn [2003] provide **trace semantics** for programs (i.e. terms) of **Kleene algebra with tests**

This is based on an elegant connection between **computation traces** in a Kripke structure and **guarded strings**

This connection **extends** very simply to the **setting of CKAT**, where

traces are related to **labeled Hasse diagrams of N-free posets**

that are associated with **guarded series-parallel strings**

As for **KAT**, a **Kripke frame** over Σ, T is a structure (K, m_K) where

K is a set of *states*, $m_K : \Sigma \to \mathcal{P}(K \times K)$ and $m_K : T \to \mathcal{P}(K)$

An **sp-trace** τ in K is essentially a **gsp-string** with the atomic guards **replaced by** states in K, such that whenever a triple $spt \in K \times \Sigma \times K$ is a subtrace of τ then $(s, t) \in m_K(p)$

As with gsp-strings, there is a **coalesced product** $\sigma \diamond \tau$ of two sp-traces σ, τ (if σ ends at the same state as where τ starts) and

a **parallel product** $\sigma || \tau$ (if σ and τ start at the same state and end at the same state)

These partial operations lift to sets X, Y of sp-traces by

- $XY = \{\sigma \diamond \tau : \sigma \in X, \tau \in Y$ and $\sigma \diamond \tau$is defined$\}$
- $X||Y = \{\sigma||\tau : \sigma \in X, \tau \in Y$ and $\sigma||\tau$is defined$\}$

Programs (terms of CKAT) are interpreted in K using the inductive definition of Kozen and Tiuryn extended by a clause for $||$:

- $[\![p]\!]_K = \{spt|(s,t) \in m_K(p)\}$ for $p \in \Sigma$
- $[\![0]\!]_K = \emptyset$ and $[\![b]\!]_K = m_K(b)$ for $b \in T$
- $[\![\bar{b}]\!]_K = K \setminus m_K(b)$ and $[\![p+q]\!]_K = [\![p]\!]_K \cup [\![q]\!]_K$
- $[\![pq]\!]_K = ([\![p]\!]_K)([\![q]\!]_K)$ and $[\![p^*]\!]_K = \bigcup_{n<\omega} [\![p]\!]_K^n$
- $[\![p||q]\!]_K = [\![p]\!]_K || [\![q]\!]_K$.

Each **sp-trace** τ has an associated **gsp-string** $\mathrm{gsp}(\tau)$ obtained by replacing every state s in τ with the corresponding **unique atomic test** $\alpha \in 2^T$ that satisfies $s \in [\![\alpha]\!]_K$

It follows that $\mathrm{gsp}(\tau)$ is the **unique guarded string** over Σ, T such that $\tau \in [\![\mathrm{gsp}(\tau)]\!]_K$

Hence the connection between **sp-trace semantics** and **gsp-strings** is the same as by Kozen and Tiuryn [2003] (the proof is also by induction on the structure of p)

Theorem 3. For a Kripke frame K, program p and sp-trace τ, we have $\tau \in [\![p]\!]_K$ if and only if $\mathrm{gsp}(\tau) \in G(p)$, whence $[\![p]\!]_K = \mathrm{gsp}^{-1}(G(p))$.

In fact gsp^{-1} is a **CKAT homomorphism** from the **free algebra** $\mathbf{C}_{\Sigma,T}$ to the algebra of **rational sets of sp-traces** over K.

The **trace model for guarded strings** has many applications since each trace in $[\![p]\!]_K$ can be interpreted as a **sequential run** of the program p starting from the first state of the trace

The sp-trace model provides a similar interpretation for a program that **forks** and **joins threads** during their runs

Each sp-trace in $[\![p]\!]_K$ is a representation of the basic programs and tests that were performed during the possibly concurrent execution of the program p

Note that there are no explicit **fork and join transitions** in an sp-trace (unlike a gsp-automaton which has to allow for nondeterministic choice)

While series-parallel traces are more complex than linear traces, they can be represented by **planar lattice diagrams**:

parallel composition is denoted by placing traces next to each other (with only one copy of the start state and end state)

sequential composition is given by placing traces vertically above each other (with only one connecting state between them).

The **sp-trace semantics** are useful for analyzing the behavior of threads that communicate only **indirectly** with other concurrent threads via **joint termination** in a single state

This is a **restricted model of concurrency**, but it has a simple **algebraic model based on Kleene algebras with tests**, and it satisfies most of the laws of **concurrent Kleene algebra**

Expanding relation algebras with concurrency

Kleene algebra with tests provides a reasonable **semantics for imperative programs**

For **specification purposes** it is useful to have the full language of **binary relations** to reason about **concurrent software**

Hence want to augment **relation algebras** with a $||$ **operation**

Recall that a ***relation algebra*** is of the form $\mathbf{A} = (A, +, 0, \wedge, \top, \bar{}, ;, 1, ^{\smile})$ where $(A, +, 0, \wedge, \top, \bar{})$ is a **Boolean algebra**, $(A, ;, 1)$ is a **monoid** and for all $x, y, z \in A$

$$x ; y \leq \bar{z} \iff x^{\smile} ; z \leq \bar{y} \iff z ; y^{\smile} \leq \bar{x}.$$

It follows that both ; and $\smile$ **distribute over the Boolean join**, and that $\smile$ is an **involution**, i.e., $x^{\smile\smile} = x$ and $(x;y)^{\smile} = y^{\smile};x^{\smile}$

Jónsson and Tarski [1951]: Every relation algebra **A** can be **embedded in a complete and atomic** relation algebra

One can define a **relational structure** on the **set of atoms** from which the algebra can be reconstructed as a **complex (powerset) algebra**

The structure is known as ***atom structure*** or ***ternary Kripke*** *frame* or ***arrow frame***, and is actually a **coalgebra**

Define an ***arrow coalgebra*** to be of the form $\gamma : X \to \mathcal{P}(X^2) \times X \times 2$ such that for all $x, y, z \in X$,

- $(x \circ y) \circ z = x \circ (y \circ z)$ where $x \circ y = \gamma_0^{-1}\{(x, y)\}$ and $A \circ z = \{a \circ z : a \in A\}$,
- $I \circ x = x = x \circ I$ where $I = \gamma_2^{-1}\{1\}$ and
- $(x, y) \in \gamma_0(z) \iff (x^\smile, z) \in \gamma_0(y) \iff (z, y^\smile) \in \gamma_0(x)$ where $x^\smile = \gamma_1(x)$.

For $A, B \subseteq X$, define $A; B = \{a \circ b : a \in A, b \in B\}$ and $A^\smile = \{a^\smile : a \in A\}$ and $1 = I$

Then the ***complex algebra*** over γ, denoted

$$\mathcal{C}m(\gamma) = (\mathcal{P}(X), \cup, \emptyset, \cap, X, ^-, ;, ^\smile, 1')$$

is a **complete relation algebra** and $;, ^\smile$ **distribute over arbitrary unions**

Expand this algebra to a **relation algebra with reflexive transitive closure** (or RAT for short) by

- $x^* = \bigcup_{n<\omega} x^n$, where $x^0 = 1'$ and $x^n = x; x^{n-1}$ for $n > 0$.

The variety generated by these algebras has a **finite equational axiomatization**, and has been studied by Tarski and Ng [1977]

Expand arrow coalgebras further by adding another factor $\mathcal{P}(X^2)$ to the type functor to interpret a **concurrency operator**

A ***concurrent arrow coalgebra*** is of the form $\gamma : X \to \mathcal{P}(X^2) \times X \times 2 \times \mathcal{P}(X^2)$ such that the projection onto the first three components is an **arrow coalgebra** and for all $x, y \in X$,

- $(x||y)||z = x||(y||z)$ and $x||y = y||x$ where $x||y = \gamma_3^{-1}\{x, y\}$
- $x \in \gamma_2^{-1}(1)$ implies $x||x = x$ and if $x \neq y$ then $x||y$ is **undefined**

The **complex algebra** of a **concurrent arrow coalgebra** is a relation algebra with an additional binary operation $||$ defined on subsets A, B of X by $A||B = \{a||b : a \in A, b \in B\}$

Adding **reflexive transitive closure** is done as before

A ***concurrent relation algebra with reflexive transitive closure*** (or **CRAT**) is an algebra of the form

$$\mathbf{A} = (A, +, 0, \wedge, \top, ^{-}, ||, ;, 1, ^{\smile}, ^{*})$$

where $\mathbf{A} = (A, +, 0, \wedge, \top, ^{-}, ;, 1, ^{\smile}, ^{*})$ is a **RAT**, $(A, +, 0, ||)$ is a **commutative semiring with zero** and $(x \wedge 1)||y = x \wedge y \wedge 1$ holds for all $x, y \in A$.

Theorem 4. The **complex algebra** of a **concurrent arrow coalgebra** is a **CRAT**, and every CRAT can be **embedded** into such a complex algebra.

A connection between **CRAT** and **CKAT**:

Theorem 5. Let $\mathbf{A} = (A, +, 0, \wedge, \top, ^{-}, ||, ;, 1, ^{\smile}, ^{*})$ be a CRAT and define $A' = \{b \in A : b \leq 1\}$. Then $\mathbf{A}'' = (A, A', +, 0, ||, \cdot, 1, ^{-}, ^{*})$ is a CKAT.

The proof is simply a matter of checking that the axioms of CKAT hold for $\mathbf{A}''$. It is currently not known if every CKAT is embeddable into an algebra of the form $\mathbf{A}''$.

The **concurrency inequality** $(x||y);(z||w) \leq (x;z)||(y;w)$ can be added to CRAT and defines a **proper subvariety**

In the language of **concurrent arrow coalgebras** the inequality takes the following form: for all $t, u, v, w, x, y, z \in X$

- $t \in u \circ v$ and $u \in x||y$ and $v \in z||w$
  $\implies \quad \exists r, s \in X$ $(t \in r||s$ and $r \in x \circ z$ and $s \in y \circ w)$

Other **inequations** that could be considered are $x||x = x$ or $x; y \leq x||y$ or $x||y \leq x; y$

Unlike Kleene algebras with tests, the **equational theory of relation algebras** is known to be **undecidable**

This is a consequence of having **complementation defined on the whole algebra**, together with the **associativity of a join-preserving operation** (Kurucz, Nemeti, Sain, Simon 1993)

Andreka, Mikulas and Nemeti [2011] show that **Kleene lattices** have **relational representations**

It is an interesting question whether this can be **extended** to **Kleene lattices with tests** or **concurrent Kleene lattices** (with tests)

Conclusion

Can **add tests to CKA** in a natural way

Extend several results from **KAT to CKAT** (completeness, trace semantics)

Can **add concurrency** to **relation algebras** with reflexive and transitive closure

Makes **concurrent composition** part of this well-known and **expressive algebraic setting**

References

Andréka, H., Mikulás, S.: Németi, I., The equational theory of Kleene lattices. Theoret. Comput. Sci. 412 (2011), no. 52, 7099–7108.

Gisher, L.: The equational theory of pomsets. Theoretical Computer Science 62 (1988) 299–224

Hoare, C. A. R., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra and its foundations. J. Log. Algebr. Program. 80 (2011), no. 6, 266–296.

Hoare, C. A. R., Möller, B., Struth, G., Wehrman, I.: Foundations of concurrent Kleene algebra. Relations and Kleene algebra in computer science, 166–186, Lecture Notes in Comput. Sci., 5827, Springer, Berlin, 2009.

Kurucz, Á., Németi, I., Sain, I., Simon, A.: Undecidable varieties of semilattice-ordered semigroups, of Boolean algebras with operators, and logics extending Lambek calculus. Logic Journal of IGPL, 1(1) (1993) 91–98.

Kozen, D.: Automata on guarded strings and applications. 8th Workshop on Logic, Language, Informations and Computation—WoLLIC'2001 (Brasília). Mat. Contemp. 24 (2003), 117–139.

Kozen, D.: On the representation of Kleene algebras with tests. Mathematical foundations of computer science 2006, 73–83, Lecture Notes in Comput. Sci., 4162, Springer, Berlin, 2006.

Kozen, D., Smith, F.: Kleene algebra with tests: completeness and decidability. Computer science logic (Utrecht, 1996), 244–259, Lecture Notes in Comput. Sci., 1258, Springer, Berlin, 1997.

Kozen, D., Tiuryn, J.: Substructural logic and partial correctness. ACM Trans. Computational Logic, 4(3) (2003) 355–378.

Lodaya, K., Weil, P.: Series-parallel languages and the bounded-width property. Theoret. Comput. Sci. 237 (2000), no. 1-2, 347–380.

Ng, K. C.: Relation Algebras with Transitive Closure. PhD thesis, University of California, Berkeley, 1984.

Ng, K. C., Tarski, A.: Relation algebras with transitive closure, Abstract 742-02-09, Notices Amer. Math. Soc. 24 (1977), A29–A30.

Pratt, V.: Modelling concurrency with partial orders. Internat. J. Parallel Prog. 15 (1) (1986) 33–71.

Thank You