Journal of Logical and Algebraic Methods in Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Concurrent Kleene algebra with tests and branching automata

Peter Jipsen[*], M. Andrew Moshier

Chapman University, Orange, CA 92866, USA

A R T I C L E   I N F O

Article history:
Received 23 November 2014
Received in revised form 26 December 2015
Accepted 28 December 2015
Available online xxxx

Keywords:
Concurrent Kleene algebra
Kleene algebra with tests
Parallel programming models
Deterministic fork-join automata
Series-parallel strings
Weak exchange law
Positive separation algebra
Flowchart schemas

A B S T R A C T

We introduce concurrent Kleene algebra with tests (CKAT) as a combination of Kleene algebra with tests (KAT) of Kozen and Smith with concurrent Kleene algebras (CKA), introduced by Hoare, Möller, Struth and Wehrman. CKAT provides a relatively simple algebraic model for reasoning about semantics of concurrent programs. We generalize guarded strings to *guarded series-parallel strings*, or gsp-strings, to give a concrete language model for CKAT. Combining nondeterministic guarded automata of Kozen with branching automata of Lodaya and Weil one obtains a model for processing gsp-strings in parallel. To ensure that the model satisfies the weak exchange law $(x\|y)(z\|w) \leq (xz)\|(yw)$ of CKA, we make use of the subsumption order of Gischer on the gsp-strings. We also define *deterministic* branching automata and investigate their relation to (nondeterministic) branching automata.

To express basic concurrent algorithms, we define concurrent deterministic flowchart schemas and relate them to branching automata and to concurrent Kleene algebras with tests.

1. Introduction

Relation algebras and Kleene algebras with tests have been used to model specifications and programs, while automata and coalgebras have been used to model state-based transition systems and object-oriented programs. Since processor speeds are leveling off, multi-core architectures and cluster-computing are becoming the norm. However there is little agreement on how to efficiently develop software for these technologies or how to model them with suitably abstract and simple principles. A main feature of using algebra is compositionality, but modeling and verifying concurrent systems in compositional ways is non-trivial because the communication mechanisms of concurrency break compositionality. The recent development of concurrent Kleene algebra [9,10,12] builds on an algebraic computational model that is well understood and has numerous applications. Hence it is useful to explore which aspects of Kleene algebras can be lifted fairly easily to the concurrent setting, and whether the simplicity of regular languages and guarded strings can be preserved along the way.

This paper is concerned with four classes of algebras and their relationships (Fig. 1): Kleene algebra (KA), Kleene algebra with tests (KAT), concurrent Kleene algebra (CKA) and the newly defined concurrent Kleene algebra with tests (CKAT). Each of them is finitely axiomatized by simple (quasi)equations based on the equational axioms of idempotent semirings. However they differ in expressive power with respect to the programming language concepts that they can express. KA is the algebraic model of regular languages which can express nondeterministic choice $+$, sequential composition $\cdot$ and finite unbounded iteration *. It also includes the two constants $0, 1$ representing the programs `abort` and `skip`. KAT

[*] Corresponding author.
E-mail addresses: jipsen@chapman.edu (P. Jipsen), moshier@chapman.edu (M.A. Moshier).

http://dx.doi.org/10.1016/j.jlamp.2015.12.005
2352-2208/© 2015 Published by Elsevier Inc.

$$\begin{tikzcd}
\text{KA} \arrow[r] \arrow[d] & \text{KAT} \arrow[d] \\
\text{CKA} \arrow[r] & \text{CKAT}
\end{tikzcd}$$

Fig. 1. Relations between classes of Kleene algebras: $\rightarrow$ adds tests, $\downarrow$ adds concurrency.

adds complementation $^{-}$ restricted to a Boolean subalgebra of tests, allowing it to express `if-then-else` and `while-do`. CKA adds concurrent composition $\|$ to KA which models the shuffle operator on strings, as well as parallel composition of pomsets and concurrent programs, while CKAT combines the features of KAT and CKA, using the signature $+, 0, \|, \cdot, 1, ^*, ^-$.

In each of the four classes of interest, a set of generators Σ represent basic (indivisible) programs or actions, the term algebra over Σ represents abstract programs (or compound actions) written in the syntax of the signature, while the free algebra shows which programs are semantically equivalent, i.e., have the same computational effect. In general it can be difficult to check this equivalence in the free algebra, but for KA and KAT the semantic models of all rational languages and all guarded rational languages give a concrete representation of the free algebra. Using automata that recognize these languages, it is possible to decide if two terms of the term algebra become identified in the free algebra, hence program equivalence is testable.

An automaton can also be viewed as an (abstract) implementation of a program (= term) in the sense that it can execute primitive commands and produce sets of traces that represent specific runs of the nondeterministic program. For KA the traces are sequences of generators, so the set of all traces is the set Σ^* of all strings with symbols from Σ. For KAT there are two types of generators: basic programs in Σ and basic tests $T = \{t_1, \dots, t_m\}$. From the basic tests a set $\Gamma = 2^T$ of guards (or atomic tests) is constructed and the relevant notion of a trace is now a *guarded string* starting and ending with a guard, and otherwise alternating basic programs with guards.

For CKA many interesting results have been obtained by Lodaya and Weil [20,21] using traces that are partially ordered multisets (or pomsets) of Pratt [23] and Gischer [7], but restricted to the class of series-parallel pomsets called sp-posets (detailed definitions are given later). This is related to the set-based traces and dependency relation used in [9,10,12] to motivate the laws of CKA. However the model of Lodaya and Weil does not satisfy the weak exchange law $(x\|y)(z\|w) \le (xz)\|(yw)$ of CKA (see Section 3 for an example). Gischer introduced a subsumption order on pomsets (recalled below) and showed that using sets of pomsets that are downward closed under this order produces a model that does satisfy the weak exchange law. Another approach is used by Hoare et al. in [11] for their Resource Model, which is a predicate transformer model that satisfies the weak exchange law. We follow Gischer's approach, with the consequence that in this model $x\|y$ means that the programs x, y are allowed to be run in parallel, but can also be run sequentially in either order.

Our aim is to investigate how guarded strings can be extended to handle concurrent composition with a similar approach as for sp-posets in [21]. The main contribution is to define a model of CKAT that satisfies the weak exchange law. In this setting the set Γ of guards is given the structure of a positive separation algebra [5] where the separating conjunction determines when two guarded series-parallel strings can be composed concurrently.

We also define a simpler notion of deterministic branching automaton. In the guarded case we extend the nondeterministic automata of Lodaya and Weil to accept guarded series parallel strings. Further we define a trace model for CKAT and give some examples of flowchart schemes to indicate how abstract programs of CKAT relate to some simple concurrent while-programs with assignments. Finally we show how the structure of a separation algebra on Γ can be used to introduce a concurrent composition on binary relations, hence giving a relational model for concurrent programs that identifies two such programs if they have the same input–output relation. The predicate transformer Resource Model of [11] mentioned above also uses a separation algebra to define the concurrent composition of two predicate transformers, but the exact relationship between our relational model and the Resource Model is a topic of future research.

2. Kleene algebra and deterministic automata

Recall that an idempotent semiring is of the form $(A, +, 0, \cdot, 1)$ such that $(A, \cdot, 1)$ is a monoid (i.e., $\cdot$ is associative $(xy)z = x(yz)$ and $x1 = x = 1x$), $(A, +, 0)$ is a (join-)semilattice with bottom (i.e., $+$ is associative, commutative $x + y = y + x$, idempotent $x + x = x$, and $x + 0 = x$) and $x(y + z) = xy + xz$, $(x + y)z = xz + yz$, $x0 = 0 = 0x$.

A *Kleene algebra* $(A, +, 0, \cdot, 1, ^*)$ is an idempotent semiring $(A, +, 0, \cdot, 1)$ with a unary operation * that satisfies the (quasi)identities $x^* = 1 + x + x^*x^*$, $xy \le y \implies x^*y \le y$ and $yx \le y \implies yx^* \le y$.

An important example of a Kleene algebra is $(\mathcal{P}(\Sigma^*), \cup, \emptyset, \cdot, \Sigma, ^*)$ where Σ is a (usually finite) set of letters, and for subsets X, Y of Σ^*, $X \cdot Y = \{vw : v \in X, w \in Y\}$, $X^0 = 1 = \Sigma$, $X^{n+1} = X \cdot X^n$ and $X^* = \bigcup_{n<\omega} X^n$. A homomorphism R is defined from the term algebra $T_{\mathrm{KA}}(\Sigma)$ to $\mathcal{P}(\Sigma^*)$ by *evaluation*, i.e.,

- $R(p) = \{p\}$ for $p \in \Sigma$, $R(0) = \emptyset$, $R(1) = \Sigma$
- $R(r + s) = R(r) \cup R(s)$, $R(r \cdot s) = R(r) \cdot R(s)$ and $R(s^*) = R(s)^*$.

A subset L of Σ^* is a *rational language* if $L = R(s)$ for some Kleene algebra term s. The subalgebra $R_\Sigma = \{R(s) : s \in T_{\mathrm{KA}}(\Sigma)\}$ is the algebra of rational languages, and the completeness theorem for Kleene algebra, due to Kozen [15], states that R_Σ is isomorphic to the free Kleene algebra $F_{\mathrm{KA}}(\Sigma)$.