Journal of Logical and Algebraic Methods in Programming 91 (2017) 17–32

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Completeness and incompleteness in nominal Kleene algebra

Dexter Kozen [a,*], Konstantinos Mamouras [b], Alexandra Silva [c]

[a] Computer Science Department, Cornell University, Ithaca, NY 14853-7501, USA
[b] CIS Department, University of Pennsylvania, Philadelphia, PA 19104-6309, USA
[c] Department of Computer Science, University College London, London WC1E 6BT, UK

ARTICLE INFO

Article history:
Received 26 January 2016
Received in revised form 9 June 2017
Accepted 11 June 2017
Available online 16 June 2017

Keywords:
Kleene algebra
Nominal sets
Programming logic

ABSTRACT

Gabbay and Ciancia (2011) presented a nominal extension of Kleene algebra as a framework for trace semantics with statically scoped allocation of resources, along with a semantics consisting of nominal languages. They also provided an axiomatization that captures the behavior of the scoping operator and its interaction with the Kleene algebra operators and proved soundness over nominal languages. In this paper, we show that the axioms proposed by Gabbay and Ciancia are not complete over the semantic interpretation they propose. We then identify a slightly wider class of language models over which they are sound and complete.

1. Introduction

Nominal sets are a convenient framework for handling name generation and binding. They were introduced by Gabbay and Pitts [1] as a mathematical model of name binding and α-conversion.

Nominal extensions of classical automata theory have recently been explored [2], motivated by the increasing need for tools for languages over infinite alphabets. These play a role in various areas, including XML document processing, cryptography, and verification. An XML document can be seen as a tree with labels from the (infinite) set of all unicode strings that can appear as attribute values. In cryptography, infinite alphabets are used as *nonces*, names used only once in cryptographic communications to prevent replay attacks. In software verification, infinite alphabets are used for references, objects, pointers, and function parameters.

In this paper, we focus on axiomatizations of the regular languages and how they can be lifted in the presence of a binding operator ν and an infinite alphabet of names. This work builds on the recent work of Gabbay and Ciancia [3], who presented a nominal extension of Kleene algebra (KA), called *nominal Kleene algebra* (NKA), as a framework for trace semantics with statically scoped allocation of resources. Gabbay and Ciancia [3] also presented an interpretation *NL* for NKA over *nominal languages* and provided a set of six axioms (Definition 2.8) that capture the behavior of the scoping operator ν and its interaction with the usual Kleene algebra operators. They showed that these axioms, in conjunction with the KA axioms (Definition 2.1), are sound over *NL*, but left open the question of completeness. In this paper we address this problem.

We first show (Theorem 4.1) that the axioms are not complete for the language model *NL* proposed by Gabbay and Ciancia. This is due to the presence of what Gabbay and Ciancia call *non-maximal planes*. The issue is rather technical, but

* Corresponding author.
E-mail addresses: kozen@cs.cornell.edu (D. Kozen), mamouras@seas.upenn.edu (K. Mamouras), alexandra.silva@gmail.com (A. Silva).
URLs: http://www.cs.cornell.edu/~kozen (D. Kozen), http://www.seas.upenn.edu/~mamouras (K. Mamouras), http://www.alexandrasilva.org (A. Silva).

http://dx.doi.org/10.1016/j.jlamp.2017.06.002
2352-2208/

$$
\begin{array}{ll}
x+(y+z)=(x+y)+z & x(yz)=(xy)z \\
x+y=y+x & x0=0x=0 \\
x+0=x+x=x & 1x=x1=x \\
x(y+z)=xy+xz & (x+y)z=xz+yz \\
1+xx^* \le x^* & 1+x^*x \le x^* \\
y+xz \le z \;\Rightarrow\; x^*y \le z & y+zx \le z \;\Rightarrow\; yx^* \le z
\end{array}
$$

Fig. 1. Axioms of Kleene algebra (KA).

we explain it in detail in §3.1 and show exactly why it causes completeness to fail. We then present in §3.2 our alternative language model *AL* that is a mild modification of *NL* and develop some basic properties of the model. We also introduce in §3.3 a variety of sound interpretations in which the scoping operator is interpreted as a summation operator over a fixed set. The axioms are not complete over these models either, but for rather uninteresting reasons. However, these models attest to the versatility of NKA.

Our main result is completeness of the axioms over *AL* (Theorem 4.2). Completeness is achieved by first transforming each expression to an equivalent expression for which only the usual Kleene algebra axioms (Definition 2.1) are needed. The steps of the transformation make use of the KA axioms along with axioms proposed by Gabbay and Ciancia (Definition 2.8) for the scoping operator. The proof is quite long but is broken into four steps: *exposing bound variables*, *scope configuration*, *canonical choice of bound variables*, and *semilattice identities*. In the last step, we make use of a technique of [4] that exploits the fact that the Boolean algebra generated by finitely many regular sets consists of regular sets and is atomic. Hence, expressions can be written as sums of atoms.

The paper is organized as follows. In §2 we recall basic material on nominal sets, Kleene algebra (KA), and nominal Kleene algebra (NKA) of Gabbay and Ciancia [3]. In §3, we discuss various interpretations: the original language model *NL* proposed in [3], our own alternative language model *AL*, and the summation models. We give a precise description of the difference between *NL* and *AL*. In §4, we present our main results on incompleteness and completeness (Theorems 4.1 and 4.2, respectively). In §5 we present concluding remarks and directions for future work.

2. Background

In this section we review basic background material on Kleene algebra (KA), nominal sets, and the nominal extension of KA (NKA) of Gabbay and Ciancia [3]. For a more thorough introduction, the reader is referred to [5,6] for nominal sets, to [7] for Kleene (co)algebra, and to [3] for NKA.

2.1. Kleene algebra (KA)

Kleene algebra is the algebra of regular expressions. Regular expressions are normally interpreted as regular sets of strings, but there are other useful interpretations: binary relation models used in programming language semantics, the (min, +) algebra used in shortest path algorithms, models consisting of convex sets used in computational geometry, and many others.

Definition 2.1 *(Kleene algebra (KA)).* A *Kleene algebra* is any structure $(K, +, \cdot, ^*, 0, 1)$ where K is a set, $+$ and $\cdot$ are binary operations on K, * is a unary operation on K, and 0 and 1 are constants, satisfying the axioms listed in Fig. 1, where we define $x \le y$ iff $x + y = y$. The top block of eleven axioms (those not involving *) are succinctly stated by saying that the structure is an idempotent semiring under $+, \cdot, 0$, and 1. The term *idempotent* refers to the axiom $x + x = x$. Due to this axiom, the ordering relation $\le$ is a partial order. The remaining four axioms involving * together say that x^*y is the $\le$-least z such that $y + xz \le z$ and yx^* is the $\le$-least z such that $y + zx \le z$.

2.2. Nominal sets

Nominal sets originated in the work of Fraenkel in 1922 and were originally used to prove the independence of the axiom of choice and other axioms. They were introduced in computer science by Gabbay and Pitts [1] as a formalism for modeling name binding in quantificational logic and the λ-calculus. Since then there has been a substantial amount of research on nominal sets in a wide variety of fields related to logic in computer science. Recent work includes the development of new programming languages [8,9] and their use in learning of automata [10].

In a nutshell, nominal sets are sets closed under the action of a certain group of symmetries $G_{\mathbb{A}}$ (defined below) and satisfying a certain finite-support condition. Nominal sets may be infinite, but the closure under symmetries makes them finitely representable and tractable for algorithms.