# Kleene Coalgebra – an overview

Alexandra Silva[1], Marcello Bonsangue[1,2], and Jan Rutten[1,3]

[1] Centrum Wiskunde & Informatica
[2] LIACS – Leiden Institute of Advanced Computer Science
[3] Radboud Universiteit Nijmegen

**Abstract.** Coalgebras provide a uniform framework for the study of dynamical systems, including several types of automata. The coalgebraic view on systems has recently been proved relevant by the development of a number of expression calculi which generalize classical results by Kleene, on regular expressions, and by Kozen, on Kleene algebra.
This note contains an overview of the motivation and results of the generic framework we developed – Kleene Coalgebra – to uniformly derive the aforementioned calculi. We present an historical overview of work on regular expressions and axiomatizations, as well a discussion of related work. We show applications of the framework to three types of probabilistic systems: simple Segala, stratified and Pnueli-Zuck.

## 1 Introduction

Computer systems have widely spread since their appearance, and they now play a crucial role in many daily activities, with their deployment ranging from small home appliances to safety critical components, such as airplane or automobile control systems. Accidents caused by either hardware or software failure can have disastrous consequences, leading to the loss of human lives or causing enormous financial drawbacks. One of the greatest challenges of computer science is to cope with the fast evolution of computer systems and to develop formal techniques which facilitate the construction of software and hardware systems.

Since the early days of computer science, many scientists have searched for suitable models of computation and for specification languages that are appropriate for reasoning about such models. When devising a model for a particular system, there is a need to encode different features which capture the inherent behavior of the system. For instance, some systems have deterministic behavior (a calculator or an elevator), whereas others have inherently non-deterministic or probabilistic behavior (think of a casino slot machine). The rapidly increasing complexity of systems demands for compositional and unifying models of computation, as well as general methods and guidelines to derive specification languages.

In the last decades, coalgebra has arisen as a prominent candidate for a mathematical framework to specify and reason about computer systems. Coalgebraic modeling works, on the surface, as follows: the basic features of a system, such as non-determinism or probability, are collected and combined in the appropriate way, determining the type of the system. This type (formally, a functor) is then used to derive a suitable equivalence relation and a universal domain of behaviors, which allow to reason about equivalence of systems. The strength of coalgebraic modeling lies in the fact that many important notions are parametrized by the type of the system. On the one hand, the coalgebraic framework is unifying, allowing for a uniform study of different systems and making precise the connection between them. On the other hand, it can serve as a guideline for the development of basic notions for new models of computation.

One of the simplest models of computation is that of a deterministic finite automaton. In his seminal paper in 1956, Kleene [32] described finite deterministic automata (which he called nerve nets), together with a specification language: regular expressions. One of his most important results is the theorem which states that any finite deterministic automaton can be characterized by a regular expression and that, conversely, every regular expression can be realized by such automaton. This theorem, which is today referred to as *Kleene's theorem*, became one of the cornerstones of theoretical computer science.

In his paper, Kleene left open the question of whether there would exist a finite, sound and complete, axiomatization of the equivalence of regular expressions, which would enable algebraic reasoning. The first answer to this question was given in 1966 by Salomaa [47], who presented two complete axiomatizations. The 1971 monograph of Conway [21] presents an extended overview of results on regular expressions and their axiomatizations. Later, in 1990, Kozen [33] showed that Salomaa's axiomatization is non-algebraic, in the sense that it is unsound under substitution of alphabet symbols by arbitrary regular expressions, and presented an algebraic axiomatization: Kleene algebras.

McNaughton and Yamada [36] gave algorithms to build a non-deterministic automaton from a regular expression and back, and introduced a notion of extended regular expression with intersection and complementation operators. This enrichment of the language of regular expressions was relevant in the context of the most important application of regular expressions at that time, in the design of digital circuits, allowing a more easy conversion of a natural language specification of problems into a regular expression.

Brzozowski [17,18] introduced the notion of derivative of regular expressions, which allowed him to prove Kleene's theorem for the extended set of regular expressions without having to recur to non-deterministic automata.

Efficient algorithms to compile regular expressions to deterministic and non-deterministic automata became crucial when regular expressions started to be widely used for pattern matching. One of the fastest (and most beautiful) algorithms to translate regular expressions into automata was devised by Berry and Sethi [8]. This algorithm became the basis of one of the first compilers of the language Esterel [7], a synchronous programming language, based on regular expressions, dedicated to embedded systems. Esterel is one of the most successful follow ups of Kleene's work: it is used as a specification language of control-intensive applications, such as the ones running in the central units of cars or airplanes. In such systems guaranteeing important safety properties is of the utmost importance and formal models of computation play a central role.

In 1981, Milner adapted Kleene and Salomaa's results to labeled transition systems: a model of computation in which non-determinism is allowed [39]. He introduced a language for finitely presented behaviors, which can be seen as a fragment of the calculus of communicating systems (CCS) [38], and a sound and complete axiomatization with respect to bisimilarity. The paper of Milner served as inspiration for many researchers in the concurrency community. Probabilistic extensions of CCS, together with sound and complete axiomatizations (with respect to the appropriate notion of equivalence) have been presented for instance in [22,23,51].

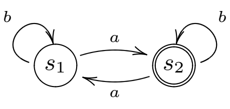
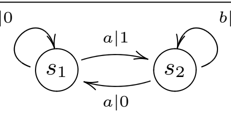
In addition to the models already mentioned, other important models of computation include Mealy machines [37] (automata with input and output), automata on guarded strings [34] and weighted automata [50]. These three models have applications, for instance, in digital circuit design, compiler optimization and image recognition, respectively.

The aim of our work is to make use of the coalgebraic view on systems to devise a framework where languages of specification and axiomatizations can be uniformly derived for a large class of systems, which include all the models mentioned above. As a sanity check, it should be possible to derive from the general framework known results. More importantly, we should be able to derive new languages and axiomatizations.

We combine the work of Kleene with coalgebra, hence the name *Kleene coalgebra.* The theory of universal coalgebra [45] provides a standard equivalence and a universal domain of behaviors, uniquely based on the type of the system, given by a functor $\mathcal{F}$. It is our main aim to show how the type of the system also allows for a uniform derivation of both a set of expressions describing the system's behavior and a corresponding axiomatization, sound and complete with respect to the equivalence induced by $\mathcal{F}$, which enables algebraic reasoning on the specifications. Furthermore, we want to show the correspondence of the behaviors denoted by the expressions in the language and the systems under study, formulating the coalgebraic analogue of Kleene's theorem.

The class of systems we study (or in other words, the class of functors $\mathcal{F}$) is large enough to cover finite deterministic automata and labeled transition systems. It includes also other models, such as Mealy machines, automata on guarded strings, weighted automata and several types of probabilistic automata, such as Segala, stratified and Pnueli-Zuck systems. From this general framework we will recover known languages and axiomatizations but, more interestingly, we will also derive new ones, for the so-called stratified and Pnueli-Zuck systems.

To give the reader a feeling of the type of expressions and axiomatizations we will derive, we show in Figure 1 a few examples of systems, together with their type and examples of valid expressions and axioms.

| Deterministic automata | Mealy machines |
|---|---|
|  |  |
| $\mathcal{F} = 2 \times \mathsf{Id}^A$ | $\mathcal{F} = (2 \times \mathsf{Id})^A$ |
| $\mu x.b(x) \oplus a(\mu y.b(y) \oplus a(x) \oplus 1)$ | $\mu x.b(x) \oplus a(\mu y.a(x) \oplus b(y)) \oplus a{\downarrow}1$ |
| $\underline{\emptyset} \oplus \varepsilon \equiv \varepsilon$ | $\mu x.\varepsilon \equiv \varepsilon[\mu x.\varepsilon/x]$ |

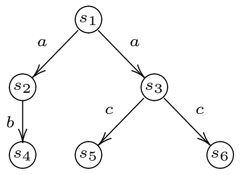
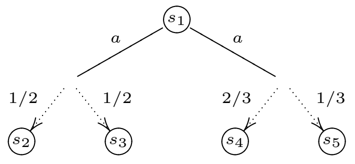
| Labeled transition systems | Segala systems |
|---|---|
|  |  |
| $\mathcal{F} = (\mathcal{P}_\omega \mathsf{Id})^A$ | $\mathcal{F} = (\mathcal{P}_\omega(\mathcal{D}_\omega \mathsf{Id}))^A$ |
| $a(\{b(\underline{\emptyset})\} \oplus \{c(\{\underline{\emptyset}\} \oplus \{\underline{\emptyset}\})\})$ | $a(\{1/2 \cdot \underline{\emptyset} \oplus 1/2 \cdot \underline{\emptyset}\}) \boxplus a(\{1/3 \cdot \underline{\emptyset} \oplus 2/3 \cdot \underline{\emptyset}\})$ |
| $\varepsilon \oplus \varepsilon \equiv \varepsilon$ | $p \cdot \varepsilon \oplus p' \cdot \varepsilon \equiv (p + p') \cdot \varepsilon$ |

**Fig. 1.** For each of the systems we show a concrete example, the corresponding functor type, an expression describing the behavior of $s_1$, and an example of a valid axiom.

## 1.1 Related work

The connection between Kleene's regular expressions, deterministic automata and coalgebra was first explored in [44,46]. Rutten studied the coalgebraic structure of the set of regular expressions, given by Brzozowski derivatives [18], in order to show that the coalgebraic semantics coincides with the standard inductive semantics of regular expressions. He proved the usefulness of the approach by proving equalities by coinduction. The coinductive proofs turned out to be, in many cases, more concise and intuitive than the alternative algebraic proof using the axioms of Kleene algebra. Later, Jacobs [31] presented a bialgebraic review on deterministic automata and regular expressions, which allowed him to present an alternative coalgebraic proof of Kozen's result on the completeness of Kleene algebras for language equivalence [33,34]. We took inspiration from all of these papers: the work of Brzozowski and Rutten led us to the definition of the coalgebraic structure on the set of expressions, whereas the work by Jacobs and Kozen served as a guideline to the proof of soundness and completeness of the axiomatization we will introduce for the set of generalized regular expressions.

In the last few years several proposals for specification languages for coalgebras appeared [40,43,30,29,20,11,12,49,35]. The languages derived in our framework (for details see the thesis of the first author [52] or the joint papers with the other authors [13,15,14,54,10,53]) are similar in spirit to that of Rössiger [43], Jacobs [30], Pattinson and Schröder [49] in that we use the ingredients of a functor for typing expressions. They differ from the logics presented in [43,30] because we do not need an explicit ”next-state” operator, as we can deduce it from the type information.

Apart from the logics introduced by Kupke and Venema in [35], the languages mentioned above do not include fixed point operators. Our language of generalized regular expressions is similar to a fragment of the logic presented in [35] and can be seen as an extension of the coalgebraic logic of [11] with fixed point operators, as well as the multi-sorted logics of [49]. However, our goal is rather different: we want (1) a finitary language that characterizes exactly all *locally finite* coalgebras; (2) a Kleene like theorem for the language or, in other words, a *map* (and not a relation) from expressions to coalgebras and vice-versa. Similar to many of the works above, we also derive a modular axiomatization, sound and complete with respect to the equivalence induced by the functor.

The languages studied in the realm of our work allow for recursive specifications and therefore formalize potentially infinite computations. This type of computations were studied also in the context of iterative theories, which have been introduced by Elgot [25]. The main example of an iterative theory is the theory of regular trees, that is trees which have finitely many distinct subtrees. Adámek, Milius and Velebil have presented Elgot's work from a coalgebraic perspective [1,2], simplified some of his original proofs, and generalized the notion of free iterative theory to any finitary endofunctor of every locally presentable category. The language associated with each functor, which we introduce in our work, modulo the axioms is closely related to the work above: it is an initial iterative algebra. This also shows the connection of our work with the work by Bloom and Ésik on iterative algebras/theories [9].

Kleene's theorem has been extended in various ways. Büchi [19] extended it to infinite words and $\omega$-automata, introducing an $\omega$ operator on languages. Ochmanski [41] introduced a concurrent version of the Kleene star operator, which lead him to define a notion of co-rational languages, obtained as the rational ones by simply replacing the star by the concurrent iteration. He then generalized Kleene's theorem showing that the recognizable trace languages are exactly the co-rational languages. Gastin, Petit and Zielonka [26,27] extended Ochmanski's results to infinite trace languages. For weighted automata, Schützenberger [50] has shown that the set of recognizable formal power series (corresponding to the behavior of weighted automata) coincides with the set of rational formal power series. For timed automata, there were several proposals, including the papers by Bouyer and Petit [16], Asarin, Caspi and Maler [4], and Asarin and Dima [5]. Recently, the results of Bouyer and Petit as well as those of Schützenberger have been extended to the class of weighted timed automata by Droste and Quaas [24]. Furthermore, Kozen has extended Kleene's language with Boolean tests as a finitary representation of regular sets of guarded strings and proved an analogue of Kleene's theorem for automata on guarded strings [34].

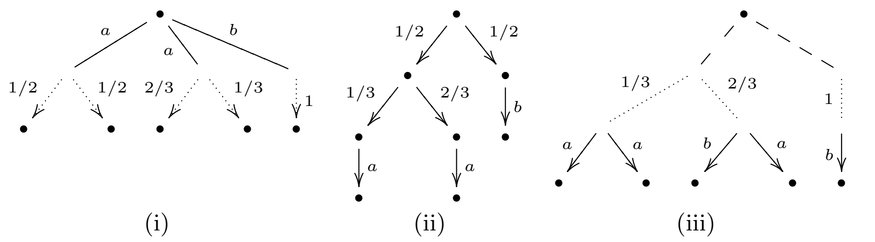


**Fig. 2.** (i) A simple Segala system, (ii) a stratified system and (iii) a Pnueli-Zuck system

From the aforementioned extensions, only the weighted automata example of Schützenberger and the automata on guarded strings of Kozen would fit in the framework we developed. The language we will derive is, however, different from the ones they proposed. Schützenberger's and Kozen's languages had full sequential composition and star in their syntax, instead of the action prefixing and unique fixed point operators that we will use in our language.

## 2 Probabilistic Systems

Many different types of probabilistic systems have been defined in literature: examples include reactive, generative, stratified, alternating, (simple) Segala, bundle and Pnueli-Zuck. Each type corresponds to a functor, and the systems of a certain type are coalgebras of the corresponding functor. A systematic study of all these systems as coalgebras was made in [6]. In particular, Figure 1 of [6] provides a full correspondence between types of systems and functors. By employing this correspondence, we will show the derived expressions and axioms for three different types of probabilistic systems: simple Segala, stratified, and Pnueli-Zuck[4].

***Simple Segala systems*** Simple Segala systems are transition systems where both probability and non determinism are present. They are coalgebras of the functor $\mathfrak{P}_\omega(\mathfrak{D}_\omega(\mathsf{Id}))^A$. Each labelled transition leads, non-deterministically, to a probability distribution of states instead of a single state. An example is shown in Figure 2(i).

For simple Segala systems we derived the following syntax and axioms.

$$\varepsilon ::= \underline{\emptyset} \mid \varepsilon \boxplus \varepsilon \mid \mu x.\varepsilon \mid x \mid a(\{\varepsilon'\}) \qquad \text{where } a \in A, p_i \in (0,1] \text{ and } \sum_{i\in 1\ldots n} p_i = 1$$

$$\varepsilon' ::= \bigoplus_{i \in 1\cdots n} p_i \cdot \varepsilon_i$$

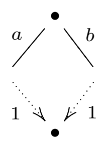
$$\begin{array}{llll}
(\varepsilon_1 \boxplus \varepsilon_2) \boxplus \varepsilon_3 \equiv \varepsilon_1 \boxplus (\varepsilon_2 \boxplus \varepsilon_3) & \varepsilon_1 \boxplus \varepsilon_2 \equiv \varepsilon_2 \boxplus \varepsilon_1 & \varepsilon \boxplus \underline{\emptyset} \equiv \varepsilon & \varepsilon \boxplus \varepsilon \equiv \varepsilon \\
(\varepsilon_1' \oplus \varepsilon_2') \oplus \varepsilon_3' \equiv \varepsilon_1' \oplus (\varepsilon_2' \oplus \varepsilon_3') & \varepsilon_1' \oplus \varepsilon_2' \equiv \varepsilon_2' \oplus \varepsilon_1' & (p_1 \cdot \varepsilon) \oplus (p_2 \cdot \varepsilon) \equiv (p_1 + p_2) \cdot \varepsilon & \\
\varepsilon[\mu x.\varepsilon/x] \equiv \mu x.\varepsilon & \gamma[\varepsilon/x] \equiv \varepsilon \Rightarrow \mu x.\gamma \equiv \varepsilon & &
\end{array}$$

[4] This is part of our joint work with Filippo Bonchi.

We show next an example of application of the axioms. The expression $a(\{1/2 \cdot \underline{\emptyset} \oplus 1/2 \cdot \underline{\emptyset}\}) \boxplus a(\{1/3 \cdot \underline{\emptyset} \oplus 2/3 \cdot \underline{\emptyset}\}) \boxplus b(\{1 \cdot \underline{\emptyset}\})$ is bisimilar to the top-most state in the simple Segala system depicted in Figure 2(i). Using the axiomatization, we can derive:

$$\begin{aligned} & a(\{1/2 \cdot \underline{\emptyset} \oplus 1/2 \cdot \underline{\emptyset}\}) \boxplus a(\{1/3 \cdot \underline{\emptyset} \oplus 2/3 \cdot \underline{\emptyset}\}) \boxplus b(\{1 \cdot \underline{\emptyset}\}) \\ \equiv\ & a(\{(1/2 + 1/2) \cdot \underline{\emptyset}\}) \boxplus a(\{(1/3 + 2/3) \cdot \underline{\emptyset}\}) \boxplus b(\{1 \cdot \underline{\emptyset}\}) \\ \equiv\ & a(\{1 \cdot \underline{\emptyset}\}) \boxplus a(\{1 \cdot \underline{\emptyset}\}) \boxplus b(\{1 \cdot \underline{\emptyset}\}) \\ \equiv\ & a(\{1 \cdot \underline{\emptyset}\}) \boxplus b(\{1 \cdot \underline{\emptyset}\}) \end{aligned}$$

Thus, we can conclude that the system presented in Figure 2(i) is bisimilar to the following one:



The language and axiomatization we presented above are the same as the one presented in [23] (with the difference that in [23] a parallel composition operator was also considered). This is of course reassuring for the correctness of the general framework we presented. In the next two examples, we will present new results (that is syntax/axiomatizations which did not exist). This is where the generality starts paying off: not only one recovers known results but also derives new ones, all of this inside the same uniform framework.

***Stratified systems*** Stratified systems are coalgebras of the functor $\mathcal{D}_\omega(\mathsf{Id}) + (\mathsf{B} \times \mathsf{Id}) + 1$. Each state of these systems either performs unlabelled probabilistic transitions or one B-labelled transition or it terminates. We first derive expressions and axioms for $\mathbb{R}^{\mathsf{Id}} + (\mathsf{B} \times \mathcal{D}_\omega(\mathsf{Id})) + 1$ and then we restrict the syntax to characterize only $\mathcal{D}_\omega(\mathsf{Id}) + (\mathsf{B} \times \mathsf{Id}) + 1$-behaviours. This, together with the introduction of some syntactic sugar, leads to the following syntax and axioms.

$$\varepsilon ::= \mu x.\varepsilon \mid x \mid \langle b, \varepsilon \rangle \mid \bigoplus_{i \in 1 \cdots n} p_i \cdot \varepsilon_i \mid \downarrow \qquad \text{where } b \in \mathsf{B}, p_i \in (0,1] \text{ and } \sum_{i \in 1 \ldots n} p_i = 1$$

$$(\varepsilon_1 \oplus \varepsilon_2) \oplus \varepsilon_3 \equiv \varepsilon_1 \oplus (\varepsilon_2 \oplus \varepsilon_3) \qquad \varepsilon_1 \oplus \varepsilon_2 \equiv \varepsilon_2 \oplus \varepsilon_1 \qquad (p_1 \cdot \varepsilon) \oplus (p_2 \cdot \varepsilon) \equiv (p_1 + p_2) \cdot \varepsilon$$
$$\varepsilon[\mu x.\varepsilon/x] \equiv \mu x.\varepsilon \qquad \gamma[\varepsilon/x] \equiv \varepsilon \Rightarrow \mu x.\gamma \equiv \varepsilon$$

Here $\downarrow$ denotes termination and $\langle b, \varepsilon \rangle$ denotes a state which can make a transition labeled by $b$ to another state specified by $\varepsilon$.

We can use these axioms (together with Kleene's theorem) to reason about the system presented in Figure 2(ii). The topmost state of this system is bisimilar to the expression

$$1/2 \cdot (1/3 \cdot \langle a, \downarrow \rangle \oplus 2/3 \cdot \langle a, \downarrow \rangle) \oplus 1/2 \cdot \langle b, \downarrow \rangle$$

which in turn is provably equivalent to $1/2 \cdot (1 \cdot \langle a, \downarrow \rangle) \oplus 1/2 \cdot \langle b, \downarrow \rangle$. That leads us to conclude that the aforementioned system is equivalent to the following simpler one.

$$\begin{tikzcd}
& \bullet \arrow[dl, "1/2"'] \arrow[dr, "1/2"] & \\
\bullet \arrow[d, "1"] & & \bullet \arrow[d, "b"] \\
\bullet \arrow[rr, "a"] & & \bullet
\end{tikzcd}$$

The language of expressions we propose for these systems is a subset of the language originally proposed in [28] (there a parallel composition operator is also considered). More interestingly, there was no axiomatization of the language in [28] and thus the axiomatization we present here is completely new.

***Pnueli-Zuck systems*** These systems are coalgebras of the functor $\mathcal{P}\mathcal{D}(\mathcal{P}(\mathsf{Id})^A)$. Intuitively, the ingredient $\mathcal{P}(\mathsf{Id})^A$ denotes $A$-labelled transitions to other states. Then, $\mathcal{D}(\mathcal{P}(\mathsf{Id})^A)$ corresponds to a probability distribution of labelled transitions and then, each state of a $\mathcal{P}\mathcal{D}(\mathcal{P}(\mathsf{Id})^A)$-coalgebra performs a non-deterministic choice amongst probability distributions of labelled transitions. For an example, consider the system depicted in Figure 2(iii).

The expressions and axioms we derive for these systems are the following.

$$\begin{array}{ll}
\varepsilon ::= \underline{\emptyset} \mid \varepsilon \boxplus \varepsilon \mid \mu x.\varepsilon \mid x \mid \{\varepsilon'\} & \text{where } a \in A, p_i \in (0,1] \text{ and } \sum_{i \in 1 \ldots n} p_i = 1 \\
\varepsilon' ::= \bigoplus_{i \in 1 \cdots n} p_i \cdot \varepsilon''_i & \\
\varepsilon'' ::= \underline{\emptyset} \mid \varepsilon'' \boxplus \varepsilon'' \mid a(\{\varepsilon\}) &
\end{array}$$

$$\begin{array}{llll}
(\varepsilon_1 \boxplus \varepsilon_2) \boxplus \varepsilon_3 \equiv \varepsilon_1 \boxplus (\varepsilon_2 \boxplus \varepsilon_3) & \varepsilon_1 \boxplus \varepsilon_2 \equiv \varepsilon_2 \boxplus \varepsilon_1 & \varepsilon \boxplus \underline{\emptyset} \equiv \varepsilon & \varepsilon \boxplus \varepsilon \equiv \varepsilon \\
(\varepsilon'_1 \oplus \varepsilon'_2) \oplus \varepsilon'_3 \equiv \varepsilon'_1 \oplus (\varepsilon'_2 \oplus \varepsilon'_3) & \varepsilon'_1 \oplus \varepsilon'_2 \equiv \varepsilon'_2 \oplus \varepsilon'_1 & (p_1 \cdot \varepsilon'') \oplus (p_2 \cdot \varepsilon'') \equiv (p_1 + p_2) \cdot \varepsilon'' & \\
(\varepsilon''_1 \boxplus \varepsilon''_2) \boxplus \varepsilon''_3 \equiv \varepsilon''_1 \boxplus (\varepsilon''_2 \boxplus \varepsilon''_3) & \varepsilon''_1 \boxplus \varepsilon''_2 \equiv \varepsilon''_2 \boxplus \varepsilon''_1 & \varepsilon'' \boxplus \underline{\emptyset} \equiv \varepsilon'' & \varepsilon'' \boxplus \varepsilon'' \equiv \varepsilon'' \\
\varepsilon[\mu x.\varepsilon/x] \equiv \mu x.\varepsilon & \gamma[\varepsilon/x] \equiv \varepsilon \Rightarrow \mu x.\gamma \equiv \varepsilon & &
\end{array}$$

The expression $\{1/3 \cdot (a(\{\underline{\emptyset}\}) \boxplus a(\{\underline{\emptyset}\})) \oplus 2/3 \cdot (b(\{\underline{\emptyset}\}) \boxplus a(\{\underline{\emptyset}\}))\} \boxplus \{1 \cdot b(\{\underline{\emptyset}\})\}$ specifies the Pnueli-Zuck system in Figure 2(iii). Note that we use the same symbol $\boxplus$ for denoting two different kinds of non-deterministic choice. This is safe, since they satisfy exactly the same axioms.

Both the syntax and the axioms we propose here for these systems are to the best of our knowledge new. In the past, these systems were studied using a temporal logic [42].

## References

1. J. Adámek, S. Milius, and J. Velebil. Free iterative theories: A coalgebraic view. *Mathematical Structures in Computer Science*, 13(2):259–320, 2003.
2. J. Adámek, S. Milius, and J. Velebil. Iterative algebras at work. *Mathematical Structures in Computer Science*, 16(6):1085–1131, 2006.
3. R. Amadio, ed, *Proceedings of FOSSACS 2008* , vol. 4962 of *LNCS*. Springer, 2008.
4. E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings of LICS 1997*, pp. 160–171, 1997.
5. E. Asarin and C. Dima. Balanced timed regular expressions. *ENTCS* 68(5), 2002.

6. F. Bartels, A. Sokolova, and E. de Vink. A hierarchy of probabilistic system types. *TCS* 327(1-2):3–22, 2004.
7. G. Berry. The foundations of Esterel. In G. Plotkin et al., eds, *Proof, Language, and Interaction*, pp. 425–454. The MIT Press, 2000.
8. G. Berry and R. Sethi. From regular expressions to deterministic automata. *TCS* 48(3):117–126, 1986.
9. S.L. Bloom and Z. Ésik. *Iteration theories: the equational logic of iterative processes.* EATCS Monographs on Theoretical Computer Science. Springer, 1993.
10. F. Bonchi, M M. Bonsangue, J. Rutten, and A. Silva. Deriving syntax and axioms for quantitative regular behaviours. In M. Bravetti et al., eds, *CONCUR*, vol. 5710 of *LNCS*, pp. 146–162. Springer, 2009.
11. M. Bonsangue and A. Kurz. Duality for logics of transition systems. In V. Sassone [48], pp. 455–469.
12. M. Bonsangue and A. Kurz. Presenting functors by operations and equations. In L. Aceto et al., eds, *FoSSaCS*, vol. 3921 of *LNCS*, pp. 172–186. Springer, 2006.
13. M. Bonsangue, J. Rutten, and A. Silva. Coalgebraic logic and synthesis of Mealy machines. In Amadio [3], pp. 231–245.
14. M. Bonsangue, J. Rutten, and A. Silva. An algebra for Kripke polynomial coalgebras. In *Proceedings LICS 2009*, pp. 49–58. IEEE, 2009.
15. M. Bonsangue, J. Rutten, and A. Silva. A Kleene theorem for polynomial coalgebras. In L. de Alfaro, ed., *FOSSACS*, vol. 5504 of *LNCS*, pp. 122–136. Springer, 2009.
16. P. Bouyer and A. Petit. A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 7(2):167–186, 2002.
17. J. A. Brzozowski. A survey of regular expressions and their applications. *IRE Transactions on Electronic Computers*, 11(0):324–335, 1962.
18. J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.
19. J. R. Buechi. On a decision method in restricted second order arithmetic. In *In Proceedings of the International Congress on Logic, Method, and Philosophy of Science*, pp. 1–12. Stanford University Press, Stanford, CA, 1962.
20. C. Cîrstea and D. Pattinson. Modular construction of modal logics. In P. Gardner et al., eds, *CONCUR*, vol. 3170 of *LNCS*, pp. 258–275. Springer, 2004.
21. J.H. Conway. *Regular algebra and finite machines.* Chapman and Hall, 1971.
22. Y. Deng and C. Palamidessi. Axiomatizations for probabilistic finite-state behaviors. In Sassone [48], pp. 110–124.
23. Y. Deng, C. Palamidessi, and J. Pang. Compositional reasoning for probabilistic finite-state behaviors. In A. Middeldorp et al., eds, *Processes, Terms and Cycles*, vol. 3838 of *LNCS*, pp. 309–337. Springer, 2005.
24. M. Droste and K. Quaas. A Kleene-Schützenberger theorem for weighted timed automata. In Amadio [3], pp. 142–156.
25. C.C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson, eds, *Logic Colloquium '73.* North-Holland Publishers, 1975.
26. P. Gastin, A. Petit, and W. Zielonka. A Kleene theorem for infinite trace languages. In J. Albert et al., eds, *ICALP*, vol. 510 of *LNCS*, pp. 254–266. Springer, 1991.
27. P. Gastin, A. Petit, and W. Zielonka. An extension of Kleene's and Ochmanski's theorems to infinite traces. *TCS* 125(2):167–204, 1994.
28. R. van Glabbeek, S. A. Smolka, and B. Steffen. Reactive, generative and stratified models of probabilistic processes. *Inf. Comput.*, 121(1):59–80, 1995.
29. R. Goldblatt. Equational logic of polynomial coalgebras. In P. Balbiani et al., eds, *Advances in Modal Logic 4*, pp. 149–184. King's College Publications, 2002.

30. B. Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *ITA*, 35(1):31–59, 2001.
31. B. Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In K. Futatsugi et al., eds, *Essays Dedicated to Joseph A. Goguen*, vol. 4060 of *LNCS*, pp. 375–404. Springer, 2006.
32. S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, pp. 3–42, 1956.
33. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of LICS 1991*, pp. 214–225, IEEE computer Society, 1991.
34. D. Kozen. Automata on guarded strings and applications. *Matemática Contemporânea*, 24:117–139, 2003.
35. C. Kupke and Y. Venema. Coalgebraic automata theory: Basic results. *Logical Methods in Computer Science*, 4(4), 2008.
36. R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, 9(0):39–47, 1960.
37. G.H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34:1045–1079, 1955.
38. R. Milner. *A Calculus of Communicating Systems*, vol. 92 of *LNCS*. Springer, 1980.
39. R. Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984.
40. L. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96, 1999.
41. E. Ochmanski. Regular behaviour of concurrent systems. *Bulletin of the EATCS*, 27:56–67, 1985.
42. A. Pnueli and L. Zuck. Probabilistic verification by tableaux. In *Proceedings LICS 1986*, pp. 322–331. IEEE Computer Society, 1986.
43. M. Rößiger. Coalgebras and modal logic. *ENTCS* 33, 2000.
44. J. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorgi et al., eds, *CONCUR*, vol. 1466 of *LNCS*, pp. 194–218. Springer, 1998.
45. J. Rutten. Universal coalgebra: a theory of systems. *TCS* 249(1):3–80, 2000.
46. J. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *TCS* 308(1-3):1–53, 2003.
47. A. Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966.
48. V. Sassone, editor. *Proceedings of FOSSACS*, vol. 3441 of *LNCS*. Springer, 2005.
49. L. Schröder and D. Pattinson. Modular algorithms for heterogeneous modal logics. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, eds, *ICALP*, vol. 4596 of *LNCS*, pp. 459–471. Springer, 2007.
50. M. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.
51. R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In B. Jonsson et al., eds, *CONCUR*, vol. 836 of *LNCS*, pp. 481–496. Springer, 1994.
52. A. Silva. *Kleene Coalgebra.* PhD thesis, Radboud Universiteit Nijmegen, 2010.
53. A. Silva, F. Bonchi, M. Bonsangue, and J. Rutten. Quantitative Kleene coalgebras. *Information and Computation*, 2011. To appear.
54. A. Silva, M. Bonsangue, and J. Rutten. Non-deterministic Kleene coalgebras. *Logical Methods in Computer Science*, 6(3), 2010.