Coalgebraic Expressions

Robert S. R. Myers
Department of Computing, Imperial College London
180 Queen's Gate, South Kensington Campus, London SW7 2AZ, United Kingdom
rm606@doc.ic.ac.uk

Abstract

We show that certain fixpoint expressions used to describe finite Kripke polynomial coalgebras can be seen as coalgebraic modal fixpoint formulae. Both the synthesis of a coalgebra from its expression and the ability to check behavioural equivalence follow from the same tableau construction. There is an associated complete equational logic, analogous to Kleene algebra, which may now be seen as an equational presentation of a fragment of the coalgebraic μ-calculus. These expressions include the regular expressions, the free Kleene algebra with tests and fragments of CCS and Linear Temporal Logic.

1 Introduction

Bonsangue, Rutten and Silva (henceforth BRS) have recently introduced certain fixpoint expressions to describe finite Kripke polynomial coalgebras [2]. The *Kripke polynomial functors* (KPF) are those endofunctors K on Set which are composed out of the identity functor, constant functors, the product and coproduct functors, the function space functor with constant domain and the finitary powerset functor. They also specify some side conditions, provided below. The *Kripke polynomial coalgebras* (KPC) are the associated K-coalgebras i.e. functions $\gamma : X \to KX$. A KPC is *finite* if its carrier set X is finite and also non-empty, a *pointed* KPC (x,γ) is a KPC γ together with a particular state $x \in X$. Examples include deterministic automata, deterministic automata on guarded strings, stream coalgebras and labelled transition systems.

To each KPF K they associate a set of fixpoint expressions Expr_K, such that every expression $\phi \in \mathsf{Expr}_K$ induces a unique finite pointed K-coalgebra $(s_\phi, \mathsf{Aut}_\phi)$ where $\mathsf{Aut}_\phi : S_\phi \to KS_\phi$ and the set S_ϕ can be thought of as the subexpressions of ϕ. They then prove that:

Every finite pointed K-coalgebra (x,γ) has an expression $\phi \in \mathsf{Expr}_K$ with $(x,\gamma) \approx (s_\phi, \mathsf{Aut}_\phi)$

where we write $(x,\gamma) \approx (x,\gamma')$ to mean that x and x' are behaviourally equivalent or *bisimilar* in the coalgebraic sense [7]. This can be seen as a generalisation of the correspondence between finite deterministic automata and the regular expressions, known as Kleene's theorem. They have also constructed a complete equational reasoning system $\equiv_K$, parametric in K. That is:

For all $\phi, \psi \in \mathsf{Expr}_K$, $(s_\phi, \mathsf{Aut}_\phi) \approx (s_\psi, \mathsf{Aut}_\psi)$ if and only if $\phi \equiv_K \psi$ is derivable.

This is analogous to the completeness of Kleene algebra: two regular expressions denote the same regular language iff their equality can be derived. Up until now, these expressions Expr_K have been seen as process algebraic, since one naturally assigns them an operational semantics. Here we show they can be seen as formulae of the coalgebraic μ-calculus [6], which is the modal μ-calculus generalised to any coalgebraic notion of transition. For every K there is an associated tableau algorithm $\mathscr{T}_K$ which may be used to decide satisfiability or dually validity of the respective coalgebraic modal fixpoint formulae. We shall discuss the following new results:

1. Each $\phi \in \mathsf{Expr}_K$ *is* a satisfiable formula of the coalgebraic μ-calculus. The satisfying model one obtains from $\mathscr{T}_K$ is precisely the automaton Aut_ϕ.

2. The equational logic $\equiv_K$ may be understood as an interesting equational presentation of a fragment of the coalgebraic μ-calculus. Then completeness of the equational system $\equiv_K$ as proved in [1] may be seen as completeness of a fragment of the coalgebraic μ-calculus.

3. To each $\phi \in \mathsf{Expr}_K$ there is a corresponding coalgebraic μ-calculus formula ϕ', such that $(s_\phi, \mathsf{Aut}_\phi)$ and $(s_\psi, \mathsf{Aut}_\psi)$ are bisimilar iff $\phi' \leftrightarrow \psi'$ is valid. This can be decided using the tableau $\mathscr{T}_K$.

4. The regular expressions, the free Kleene algebra with tests and parts of CCS and LTL all arise as fragments. We therefore obtain natural algorithms for both their synthesis and for the testing of their behavioural equivalence in a purely generic manner.

When testing behavioural equivalence, the μ-calculus formulae we consider never contain interleaving μs and νs, simplifying things considerably. Although we do not discuss complexity here, we expect our generic decision procedure to yield e.g. PSPACE-complete algorithms for testing equivalence of regular expressions and the free Kleene algebra with tests.

2 Kripke Polynomial Functors

The Kripke polynomial functors $K : \mathsf{Set} \to \mathsf{Set}$ are inductively defined:

$$K ::= Id \mid B \mid K + K \mid K \times K \mid K^A \mid \mathscr{P}_\omega K$$

- $Id : \mathsf{Set} \to \mathsf{Set}$ is the identity functor.
- $B : \mathsf{Set} \to \mathsf{Set}$ is a constant functor such that the set B is finite. We also assume that B is a join-semilattice with a bottom element. This means that B is equipped with a binary operation $\vee : B \times B \to B$ which is associative, commutative and idempotent and also a constant $\bot \in B$ with $b \vee \bot = b$ for all $b \in B$. This is the assumption made by BRS and we shall see that it is very natural. In fact every *finite* join-semilattice with a bottom element is also a lattice because it has all joins and hence all meets.
- $+ : \mathsf{Set}^2 \to \mathsf{Set}$ is defined $X + Y = \{(1,x) : x \in X\} \cup \{(2,y) : y \in Y\} \cup \{\bot, \top\}$ and if $f : X \to X'$ and $g : Y \to Y'$ then $f + g : X + Y \to X' + Y'$ is defined in the normal way, where additionally $f + g(\bot) = \bot$ and $f + g(\top) = \top$. This abnormal definition may be understood as forcing the coproduct to be a functor on the category of join-semilattices with bottom, see below.
- $\times : \mathsf{Set}^2 \to \mathsf{Set}$ is the standard Cartesian product functor.
- $(-)^A$ is the function space functor with constant finite domain A i.e. X^A is the set of functions from A to X and if $f : X \to Y$ then $f^A : X^A \to Y^A$ is defined $f^A(\alpha) = f \circ \alpha$.
- $\mathscr{P}_\omega$ is the finitary powerset functor with $\mathscr{P}_\omega X = \{A \subseteq X : A \text{ finite}\}$ and if $f : X \to Y$ then $\mathscr{P}_\omega f : \mathscr{P}_\omega X \to \mathscr{P}_\omega Y$ is the direct image of f, restricted to finite subsets of X.

Example 1. *We consider four examples:*

1. *Let $2 = \{0,1\}$ denote the minimal Boolean lattice. Then $\mathsf{DA} = 2 \times Id^A$ is the deterministic automata functor with label set A. DA-coalgebras $\gamma : X \to \mathsf{DA}X$ are precisely deterministic automata: each γ may be understood as an output function $out_\gamma : X \to 2$ where $out_\gamma = \pi_1 \circ \gamma$ and transition function $trans_\gamma : X \to X^A$ where $trans_\gamma = \pi_2 \circ \gamma$.*

2. *Let $\mathsf{Tests} = \{t_1, \ldots, t_n\}$ be a finite set, whose elements are thought of as* tests. *We may think of the set $BA_{\mathsf{Tests}} = \mathscr{P}\mathscr{P}\mathsf{Tests}$ as the free Boolean lattice over* Tests. *Then $\mathsf{AGS} = BA_{\mathsf{Tests}} \times Id^{\mathsf{Atoms} \times A}$ is the automata on guarded strings functor with tests* Tests, *label set A and $\mathsf{Atoms} = \mathscr{P}\mathsf{Tests}$. The AGS-coalgebras are precisely the deterministic automata on guarded strings [5], with parameters* Tests *and A. Finally let $out_\gamma^{\mathsf{Tests}} = \pi_1 \circ \gamma$ and $trans_\gamma^{\mathsf{Tests}} = \pi_2 \circ \gamma$.*

3. *Let $Prop = \{p_1,\dots,p_n\}$ be some finite set of propositional variables and $Prop_L = \mathscr{P}Prop$ be the powerset lattice. Then we call* $\mathsf{Str} = Prop_L \times Id$ *the $Prop_L$-stream functor.* Str*-coalgebras are streams over the lattice $Prop_L$ and the behaviour (i.e. corresponding element of the* Str*-final coalgebra [7]) of every finite* Str*-coalgebra is the infinite word $r(s)^{\omega}$ where $r,s \in \mathscr{P}Prop_L{}^*$ and s is not the empty word. Let $head_\gamma = \pi_1 \circ \gamma$ and $tail_\gamma = \pi_2 \circ \gamma$.*

4. *Fix some finite set Act of actions, then* $\mathsf{LTS} = (\mathscr{P}_\omega Id)^A$ *is the labelled transition systems functor and* LTS*-coalgebras are labelled transition systems.*

Each KPF can also be understood as a functor on the category $\mathsf{JSL}_\perp$ of join-semilattices with bottom and semilattice morphisms that preserve the bottom. Recall each join-semilattice $(X, \vee_X)$ has a natural partial ordering $x \leq_X y \iff x \vee y = y$, the bottom is the least element with respect to this ordering. We only provide their action on objects:

- The identity functor maps join-semilattices with bottom to themselves.
- The constant functor B maps to the particular join-semilattice with bottom B, by assumption.
- The coproduct $(X+Y, \vee_+, \perp_+)$ is defined $(1,x) \vee_+ (2,y) = \top_+$, $(1,x) \vee_+ (1,x') = (1, x \vee_X x')$, $(2,y) \vee_+ (2,y') = (2, y \vee_Y y')$. The element $\perp_+ \in X+Y$ is required to be the bottom.
- $(X \times Y, \vee_{X\times Y}, \perp_{X\times Y})$ is defined $\perp_{X\times Y} = (\perp_X, \perp_Y)$ and $(x,y) \vee_{X\times Y} (x',y') = (x \vee_X x', y \vee_Y y')$.
- $(X^A, \vee_{X^A}, \perp_{X^A})$ is defined $\perp_{X^A} = \lambda a.\perp_X$ and $f \vee_{X^A} g = \lambda a.(f(a) \vee_X g(a))$.
- $(\mathscr{P}_\omega X, \vee_{\mathscr{P}_\omega X}, \perp_{\mathscr{P}_\omega X})$ is defined $\perp_{\mathscr{P}_\omega X} = \emptyset$ and $A \vee_{\mathscr{P}_\omega X} B = A \cup B$.
- Functors are closed under composition, thus every KPF can be seen as an endofunctor on $\mathsf{JSL}_\perp$.

3 Generic Syntax

We now introduce the syntax used by BRS. It differs slightly in that we have used different labels for the symbols and avoided using a type system. Syntactically we can view any KPF K as its parse tree. We think of each node of the tree as being labelled by both the respective component functor C and also a positive integer n, this being the step at which the node is visited in a depth-first left-child first search. We denote the resulting tree $Tree_K$, examples of the construction for DA and LTS are provided below.

$$\begin{tikzcd}
& \times : \mathbf{1} \arrow[dl] \arrow[dr] & & (-)^A : \mathbf{1} \arrow[d] \\
2 : \mathbf{2} & & (-)^A : \mathbf{3} \arrow[d] & \mathscr{P}_\omega : \mathbf{2} \arrow[d] \\
& & Id : \mathbf{4} & Id : \mathbf{3}
\end{tikzcd}$$

Definition 1. *The expressions* Expr_K *associated with a KPF K are defined in terms of collection of interleaved grammars, using the structure of $Tree_K$. Let $\mathfrak{s}_1$, $\mathfrak{s}_2$ be arbitrary subtrees of $Tree_K$:*

$$\begin{array}{ll}
\mathscr{L}_{Id:1} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 & \mathscr{L}_{B:n} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [b] \quad (b \in B) \\
\mathscr{L}_{\mathfrak{s}_1 + :n \, \mathfrak{s}_2} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [e_1]\psi_1 \mid [e_2]\psi_2 & \mathscr{L}_{\mathfrak{s}_1 \times :n \, \mathfrak{s}_2} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi_1 \mid [\pi_2]\psi_2 \\
(a \in A) \quad \mathscr{L}_{(\mathfrak{s}_1)^A:n} \ni \phi ::= \top \mid \phi_2 \wedge \phi_2 \mid \langle a \rangle \psi_1 & \mathscr{L}_{\mathscr{P}_\omega:n(\mathfrak{s}_1)} \ni \phi ::= \top \mid \phi_2 \wedge \phi_2 \mid \Diamond \psi_1
\end{array}$$

where $\psi_i \in \mathscr{L}_{\mathfrak{s}_i}$ for $i = 1,2$. The expressions Expr_K *are the closed and guarded members of $\mathscr{L}^{\nu}_{Tree_K}$:*

$$\begin{array}{c}
\mathscr{L}^{\nu}_{Tree_K} \ni \phi ::= \mathscr{L}_{Tree_K} \mid x \mid \nu x.\phi \\
\textit{moreover} \quad \mathscr{L}_{Id:n} := \mathscr{L}^{\nu}_{Tree_K} \quad \textit{if } n > 1
\end{array}$$

The language $\mathscr{L}^{\nu}_{Tree_K}$ is the minimal grammar containing $\mathscr{L}_{Tree_K}$, all variables x from some countably infinite set $Var = \{x_1, x_2, \ldots\}$ and closed under νx-prefixing. An expression is closed and guarded if every variable x appears in the scope of some νx and variables are separated from their binders by some modal operator $\heartsuit$, in the usual way. Note that Expr_K is inductively defined in terms of all the languages $\mathscr{L}_{\mathfrak{s}}$ where $\mathfrak{s}$ is a subtree of $Tree_K$.

Example 2. 1. *Expressions for finite deterministic automata where $a \in A$:*

$$\mathsf{Expr}_{\mathsf{DA}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi \mid [\pi_2]\chi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$

$$\mathscr{L}_{2:2} \ni \psi ::= \top \mid \psi_1 \wedge \psi_2 \mid [0] \mid [1] \qquad \mathscr{L}_{(Id:4)^A:3} \ni \chi ::= \top \mid \chi_1 \wedge \chi_2 \mid \langle a \rangle \phi$$

In terms of regular expressions, $\mathbf{0} = [\pi_1][0]$ should be thought of as the empty language, $\mathbf{1} = [\pi_1][1]$ as the empty word ε and $\mathbf{a}.\phi = [\pi_2]\langle a \rangle \phi$ as prefixing by the letter a. Later we show the regular expressions arise as a fragment. Using these abbreviations we obtain a single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{DA}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \mathbf{0} \mid \mathbf{1} \mid \mathbf{a}.\phi \mid x \mid \nu x.\phi$$

2. *Expressions for finite deterministic automata on guarded strings where $b \in BA_{\mathsf{Tests}} = \mathscr{P}\mathscr{P}\mathsf{Tests}$, $A \in \mathsf{Atoms} = \mathscr{P}\mathsf{Tests}$ and $a \in A$:*

$$\mathsf{Expr}_{\mathsf{AGS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi \mid [\pi_2]\chi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$

$$\mathscr{L}_{BA_{\mathsf{Tests}}:2} \ni \psi ::= \top \mid \psi_1 \wedge \psi_2 \mid [b] \qquad \mathscr{L}_{(Id:4)^{\mathsf{Atoms} \times A}:3} \ni \chi ::= \top \mid \chi_1 \wedge \chi_2 \mid \langle (A,a) \rangle \phi$$

Because BA_{Tests} is the free Boolean algebra generated by Tests, instead of elements $b \in BA_{\mathsf{Tests}}$ we may instead use $\mathsf{Prop}(\mathsf{Tests})$, the propositional formulae β with variables in Tests. More explicitly let $\sigma : \mathsf{Prop}(\mathsf{Tests}) \to BA_{\mathsf{Tests}}$ be the unique extension of the valuation $\sigma(t_i) = \{A \in \mathsf{Atoms} : t_i \in A\}$ for $i = 1, \ldots, n$. We define $\langle \beta \rangle = [\pi_1][\beta]$ and $[\beta \to a]\phi = \bigwedge_{A \in \sigma(\beta)} [\pi_2]\langle (A,a) \rangle \phi$, so that a guarded string [5] $A_1 a_1 \ldots A_{n-1} a_n A_n \in \mathsf{Atoms}(A \cdot \mathsf{Atoms})^*$ arises as the formula $[A_1 \to a_1] \ldots [A_{n-1} \to a_n]\langle A_n \rangle$. This yields the single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{AGS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle \beta \rangle \mid [\beta \to a]\phi \mid x \mid \nu x.\phi$$

3. *Expressions for finite Str-coalgebras where $P \in Prop_L$ i.e. P is a subset of Prop:*

$$\mathsf{Expr}_{\mathsf{Str}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid [\pi_1]\psi \mid [\pi_2]\phi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$
$$\mathscr{L}_{Prop_L:2} \ni \psi := \top \mid \psi_1 \wedge \psi_2 \mid [P]$$

We have the Next modality $\bigcirc\phi = [\pi_2]\phi$ and the Always modality $\Box\phi = \nu x.(\phi \wedge \bigcirc x)$ of LTL. Also for $P \subseteq Prop$ let $\langle P \rangle = [\pi_1][P]$. Again we have a single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{Str}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle P \rangle \mid \bigcirc\phi \mid x \mid \nu x.\phi$$

4. *Expressions for finite LTS-coalgebras where $a \in Act$:*

$$\mathsf{Expr}_{\mathsf{LTS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle \psi \mid x \mid \nu x.\phi \qquad (\phi \textit{ closed and guarded})$$

$$\mathscr{L}_{\mathscr{P}_\omega:2(Id:3)} \ni \psi ::= \top \mid \psi_1 \wedge \psi_2 \mid \Diamond\phi$$

For each $a \in Act$ we let $\langle \mathbf{a} \rangle \phi = \langle a \rangle \Diamond \phi$ this being the standard relational diamond. Again we have a single-sorted closed and guarded fragment:

$$\mathsf{Expr}^1_{\mathsf{LTS}} \ni \phi ::= \top \mid \phi_1 \wedge \phi_2 \mid \langle \mathbf{a} \rangle \phi \mid x \mid \nu x.\phi$$

The above languages are almost exactly the same as BRS's syntax, to obtain their languages one may bijectively relabel our symbols as follows:

$$\top \to_\theta \emptyset \qquad \phi \wedge \psi \to_\theta \phi + \psi \qquad \nu x.\phi \to_\theta \mu x.\phi$$
$$[e_1]\phi \to_\theta l[\phi] \quad [e_2]\phi \to_\theta r[\phi] \quad [\pi_1]\phi \to_\theta l(\phi) \quad [\pi_2]\phi \to_\theta r(\phi) \quad \langle a\rangle\phi \to_\theta \langle a\rangle\phi \quad \Diamond\phi \to_\theta \{\phi\}$$

The top line is rather interesting:

- The join-semilattice structure $+$ and $\emptyset$ – familiar from the regular expressions – corresponds with conjunction and $\top$. Indeed the latter also form a join-semilattice structure via propositional logical equivalence.
- Notice that to build an automaton for the regular expression $r+s$ one must build automata for *both* r and s, in this sense $+$ is a conjunction.
- To understand why $\emptyset$ corresponds with $\top$, recall that the minimal deterministic automaton that accepts the empty language is the single non-final state x where all transitions $a \in A$ loop back into x. Then ask: what is the minimal automaton which models the formula $\top$? There are two models consisting of only one state: one will accept the empty language while the other accepts its complement. In fact the former is the more natural choice because it corresponds with the bottom element of the final DA-coalgebra, which inherits its join-semilattice structure from the fact that DA is a functor on $\mathsf{JSL}_\perp$.
- Finally notice μ actually maps to ν. It is understandable that BRS decided to use μ because of the strong analogy with the Kleene star, which is best thought of as a least fixpoint and is axiomatised as such in Kleene algebra [4]. However, recall that finite deterministic automata require loops in order to accept regular languages with stars, and loops – being infinite behaviours – are represented by νs not μs.

We chose the relabellings in the second line to emphasise that the various unary operators are *modal* operators. In fact each of them is a well-known coalgebraic modal operator and the symbols we have chosen for them are standard. The multisorted languages Expr_K above and also the 'glued together' single-sorted language Expr^1_K is actually a specific application of a general construction in coalgebraic modal logic, where one builds the language and logic of composite functors out the languages and logics of their components [8].

4 Generic Final Semantics

For each K, BRS inductively define a coalgebra $\lambda_K : \mathsf{Expr}_K \to K(\mathsf{Expr}_K)$ over the expressions, which yields a finite pointed coalgebra or *automaton* $(s_\phi, \mathsf{Aut}_\phi)$ for each expression $\phi \in \mathsf{Expr}_K$. In this section we present an equivalent construction using tableau rules, whose meaning will be explained in the following section. We proceed informally since we lack the space for a detailed account.

A *sequent* Γ for $\phi \in \mathsf{Expr}_K$ is a finite subset of either Expr_K or $\mathscr{L}_\mathfrak{s}$ for some subtree $\mathfrak{s}$ of $Tree_K$. It should be thought of as the conjunction of its elements. Note it must be a finite subset of a particular component language: they may not be mixed. The following rules relate a single sequent (the premise) to one or more sequents (the conclusions).

$$(\wedge)\ \frac{\{\phi\wedge\psi\}\cup\Gamma}{\{\phi,\psi\}\cup\Gamma} \qquad (\nu)\ \frac{\{\nu x.\phi\}\cup\Gamma}{\{\phi[x:=\nu x.\phi]\}\cup\Gamma} \qquad (e_i)\ \frac{\{[e_i]\phi_1,\dots,[e_i]\phi_n\}\cup\Gamma}{\{\phi_1,\dots,\phi_n\}} \qquad (\Diamond)\ \frac{\{\Diamond\phi_1,\dots,\Diamond\phi_n\}\cup\Gamma}{\{\phi_1\}\quad\dots\quad\{\phi_n\}}$$

$$(\pi)\ \frac{\{[\pi_1]\phi_1,\dots,[\pi_1]\phi_m,[\pi_2]\psi_1,\dots,[\pi_2]\psi_n\}\cup\Gamma}{\{\phi_1,\dots,\phi_m\}\qquad\{\psi_1,\dots,\psi_n\}} \qquad (\langle a_1\dots a_m\rangle)\ \frac{\{\langle a_1\rangle\phi_1^1,\dots,\langle a_1\rangle\phi_1^{n_1},\dots,\langle a_m\rangle\phi_m^1,\dots,\langle a_m\rangle\phi_m^{n_m}\}\cup\Gamma}{\{\phi_1^1,\dots,\phi_1^{n_1}\}\quad\dots\quad\{\phi_m^1,\dots,\phi_m^{n_m}\}}$$

There are various conditions on how these rules are applied but we only provide them informally here. Start with the single node or sequent $\{\phi\}$ and then apply a rule whose premise can be written as $\{\phi\}$, repeating this matching process on each of the rule's conclusions. When choosing a rule to apply, the rules $(\wedge)$ and (ν) take precedence over the others. When no rule may be applied or when we come across a sequent we have already seen, we don't match rules to that sequent. The repetition of a sequent correspond with a *loop* in the automaton if the first occurrence of the sequent appears higher in the derivation tree. Roughly speaking, the process terminates because there are only finitely many possible sequents, namely the finite subsets of the subexpressions of ϕ.

Example 3. *We provide three examples of tableau:* $\top$, $\mathbf{a}.\mathbf{1}\wedge\mathbf{b}.\mathbf{1}$ *and* $\mathbf{a}.\nu x.(\mathbf{1}\wedge\mathbf{a}.x)$ *from* $\mathsf{Expr}^1_{\mathsf{DA}}$*. Recall that* $\mathbf{1} := [\pi_1][1]$ *and* $\mathbf{a}.\phi := [\pi_2][a]\phi$ *in* $\mathsf{Expr}_{\mathsf{DA}}$.

$$\{\top\} \qquad \dfrac{\dfrac{\{[\pi_2]\langle a\rangle[\pi_1][1]\wedge[\pi_2]\langle b\rangle[\pi_1][1]\}}{\{[\pi_2]\langle a\rangle[\pi_1][1],[\pi_2]\langle b\rangle[\pi_1][1]\}}\,(\wedge)}{\emptyset \qquad \dfrac{\{\langle a\rangle[\pi_1][1],\langle b\rangle[\pi_1][1]\}}{\dfrac{\{[\pi_1][1]\}}{\{[1]\}\qquad\emptyset}\,(\pi) \qquad (\{[\pi_1][1]\})}\,(\langle ab\rangle)}\,(\pi) \qquad \dfrac{\{[\pi_2]\langle a\rangle(\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x))\}}{\emptyset \qquad \dfrac{\{\langle a\rangle(\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x))\}}{\dfrac{\{\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\}}{\dfrac{\{[\pi_1][1]\wedge[\pi_2]\langle a\rangle\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\}}{\dfrac{\{[\pi_1][1],[\pi_2]\langle a\rangle\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\}}{\{[1]\}\qquad(\{\langle a\rangle\nu x.([\pi_1][1]\wedge[\pi_2]\langle a\rangle x)\})}\,(\pi)}\,(\wedge)}\,(\nu)}\,(\langle a\rangle)}\,(\pi)$$

Sequents are enclosed in brackets $(\cdot)$ if they already appear earlier in the construction.

Each expression ϕ yields a unique tableau which in turn completely determines the automaton $(s_\phi, \mathsf{Aut}_\phi)$. Again we will not explicitly provide the process used to convert the tableau to an automaton, although it is not complicated and is of course generic in K. Instead we provide the deterministic automata correspondents of the tableaux above, where final states are enclosed in a double box and we assume $A = \{a,b\}$:

$$\begin{tikzcd}
\{\top\} \arrow[loop right, "{a,b}"] & \{\mathbf{a}.\mathbf{1}\wedge\mathbf{b}.\mathbf{1}\} \arrow[r, "{a,b}"] & \boxed{\boxed{\{\mathbf{1}\}}} \arrow[r, "{a,b}"] & \{\top\} \arrow[loop right, "{a,b}"] & \{\mathbf{a}.\nu x.(\mathbf{1}\wedge\mathbf{a}.x)\} \arrow[r, "a"] \arrow[dr, "b"'] & \boxed{\boxed{\{\mathbf{1},\mathbf{a}.\nu x.(\mathbf{1}\wedge\mathbf{a}.x)\}}} \arrow[loop above, "a"] \arrow[d, "b"'] \\
 & & & & & \{\top\} \arrow[loop right, "{a,b}"]
\end{tikzcd}$$

Note that each automaton has a 'sink-node' $\{\top\}$ which corresponds with the empty-language. Notice also that the right-most automaton has a loop on $\{\mathbf{1},\mathbf{a}.\nu x.(\mathbf{1}\wedge\mathbf{a}.x)\}$, this corresponds with the duplication of that same sequent in the tableau above and the fact that it occurred higher-up in the tableau. A duplication also occurs in the middle tableau but because the original isn't higher we don't get a loop. In fact these are automata for the regular expressions: $\emptyset$, $a+b$ and aa^*.

When BRS construct the automaton for an expression they first inductively define $\lambda_K : \mathsf{Expr}_K \to K(\mathsf{Expr}_K)$ and then show how to construct an automaton from it: this requires checking for loops to ensure termination. We have briefly outlined a method where one first constructs a tableau and then converts it to an automaton. Their process can then be understood as converting the tableau to an automaton *as the tableau is being built* i.e. it composes the two steps we have sketched. Thus the generalisation of Kleene's theorem follows:

Theorem 1. *[1] For every finite pointed KPC (x,γ) with $\gamma : X \to KX$ there exists an expression $\phi \in \mathsf{Expr}_K$ with $(\{\phi\},Aut_\phi\}$ bisimilar to (x,γ)*

Proof. Follows because the above automaton construction turns out to be equivalent to BRS's and they provide an algorithm which converts any finite pointed KPC into an expression ϕ satisfying the above property. Alternatively it is possible to obtain the result in terms of join-semilattices with bottom and the semantics presented in the next section. $\square$

5 Generic Modal Semantics and Equational Logic

For each of our four example functors $K \in \{\mathsf{DA}, \mathsf{AGS}, \mathsf{Str}, \mathsf{LTS}\}$ and any finite coalgebra $\gamma : X \to KX$ we provide a semantics $\models^{\gamma}_{K} \subseteq X \times \mathsf{Expr}^1_K$ for the single-sorted language Expr^1_K. In fact this is the glueing together of a more general multisorted semantics. Also recall that for any join-semilattice $(B, \vee_B)$ there is a natural partial ordering $x \leq_B y \iff x \vee_B y = y$.

$$x \models^{\gamma}_{K} \top \; always \qquad x \models^{\gamma}_{K} \phi_1 \wedge \phi_2 \iff x \models^{\gamma}_{K} \phi_i \; for \; i = 1,2$$
$$x \models^{\gamma}_{K} \nu x.\phi \iff \forall n \in \omega . x \models^{\gamma}_{K} \phi[x := \nu x.\phi]^n[x := \top]$$
$$x \models^{\gamma}_{\mathsf{DA}} \mathbf{a}.\phi \iff trans_\gamma(x)(a) \models^{\gamma}_{\mathsf{DA}} \phi \qquad x \models^{\gamma}_{\mathsf{DA}} \mathbf{0} \iff out_\gamma(x) \geq_2 0 \qquad x \models^{\gamma}_{\mathsf{DA}} \mathbf{1} \iff out_\gamma(x) \geq_2 1$$
$$x \models^{\gamma}_{\mathsf{AGS}} [\beta \to a]\phi \iff \forall A \in \sigma(\beta). trans^{\mathsf{Tests}}_\gamma(A,a) \models^{\gamma}_{\mathsf{AGS}} \phi \qquad x \models^{\gamma}_{\mathsf{AGS}} \langle \beta \rangle \iff out^{\mathsf{Tests}}_\gamma(x) \geq_{BA} \sigma(\beta)$$
$$x \models^{\gamma}_{\mathsf{Str}} \bigcirc \phi \iff tail_\gamma(x) \models^{\gamma}_{\mathsf{Str}} \phi \qquad x \models^{\gamma}_{\mathsf{Str}} \langle P \rangle \iff head_\gamma(x) \geq_{Prop_L} P$$
$$x \models^{\gamma}_{\mathsf{LTS}} \langle \mathbf{a} \rangle \phi \iff \exists y \in \gamma(x)(a). y \models^{\gamma}_{\mathsf{LTS}} \phi$$

As usual we say a state x in a K-coalgebra γ *satisfies* $\phi \in \mathsf{Expr}^1_K$ if $x \models^{\gamma}_{K} \phi$ and that ϕ is *valid* if every state of every K-coalgebra satisfies ϕ. For example, in the three deterministic automata above each state Γ satisfies every $\phi \in \Gamma$. Notice that the semantics for LTS is exactly that of the modal μ-calculus with ν and $\wedge$ but without the duals μ and $\vee$. In the same way, for each K the semantics $\models^{\gamma}_{K}$ is precisely the semantics of the *coalgebraic μ-calculus* [6], which is the natural generalisation of the modal μ-calculus. More precisely, each unary and nullary modal operator $[b], [e_i], [\pi_i], \langle a \rangle, \Diamond$ can be naturally assigned a coalgebraic semantics which induces the above semantics.

The tableau construction of the previous section turns out to be a slight rewriting of the tableau construction used in the coalgebraic μ-calculus.

Theorem 2. *For all $\phi \in \mathsf{Expr}_K$, BRS's automata construction is a tableau construction which builds a satisfying model of ϕ, seen as a formula of the respective coalgebraic μ-calculus*

In [2] BRS define a complete equational logic $\equiv_K \subseteq \mathsf{Expr}^1_K \times \mathsf{Expr}^1_K$[1], generic in K. By *completeness* we mean $(\{\phi\}, \mathsf{Aut}_\phi)$ and $(\{\psi\}, \mathsf{Aut}_\psi)$ are bisimilar iff $\phi \equiv_K \psi$ is derivable. For our example functors their equational logic takes the following form:

$$\top \wedge \phi \equiv_K \phi \qquad \phi \wedge \psi \equiv_K \psi \wedge \phi \qquad \phi \wedge (\psi \wedge \chi) \equiv_K (\phi \wedge \psi) \wedge \chi$$
$$\nu x.\phi \equiv_K \phi[x := \nu x.\phi] \qquad \phi[x := \psi] \wedge \psi \equiv_K \psi \implies (\nu x.\phi) \wedge \psi \equiv_K \psi$$
$$\top \equiv_{\mathsf{DA}} \mathbf{a}.\top \qquad \top \equiv_{\mathsf{DA}} \mathbf{0} \qquad \mathbf{0} \wedge \mathbf{1} \equiv_{\mathsf{DA}} \mathbf{1} \qquad \mathbf{a}.\phi \wedge \mathbf{a}.\psi \equiv_{\mathsf{DA}} \mathbf{a}.(\phi \wedge \psi)$$
$$\top \equiv_{\mathsf{AGS}} [\beta \to a]\top \qquad \top \equiv_{\mathsf{AGS}} [\bot \to a]\phi \qquad \top \equiv_{\mathsf{AGS}} \langle \bot \rangle \qquad \langle \beta \rangle \wedge \langle \beta' \rangle \equiv_{\mathsf{AGS}} \langle \beta \vee \beta' \rangle$$
$$[\beta \to a]\phi \wedge [\beta' \to a]\phi \equiv_{\mathsf{AGS}} [\beta \vee \beta' \to a]\phi \qquad [\beta \to a]\phi \wedge [\beta \to a]\psi \equiv_{\mathsf{AGS}} [\beta \to a](\phi \wedge \psi)$$
$$\top \equiv_{\mathsf{Str}} \bigcirc \top \qquad \top \equiv_{\mathsf{Str}} \langle \emptyset \rangle \qquad \langle P \rangle \wedge \langle P' \rangle \equiv_{\mathsf{Str}} \langle P \cup P' \rangle \qquad \bigcirc \phi \wedge \bigcirc \psi \equiv_{\mathsf{Str}} \bigcirc(\phi \wedge \psi)$$

Also $\equiv_K$ is a congruence for each unary modal operator $\mathbf{a}.$, $[\beta \to a]$ and $\bigcirc$ and satisfies the usual axioms of equational logic, plus uniform renaming of variables and their binders. Interestingly, the equations involving νx are precisely those used by Kozen in [3], to prove the completeness of the aconjunctive fragment of the modal μ-calculus. There he writes them in their dual form, using $\vee$ and μ. Aside from the semilattice equations for $\wedge$ and $\top$ and renaming of variables, the other equations may be be understood as the rank-1 modal formulae [9] which only involve $\wedge$ and $\top$. There are no equations for $\mathscr{P}_\omega$ involving modal operators precisely because $\Diamond$ doesn't preserve conjunctions.

6 Generic Decidability and a Translation

We mentioned in the introduction that for every expression $\phi \in \mathsf{Expr}_K$ there is an associated expression ϕ' such that: $\phi \equiv_K \psi$ if and only if $\phi' \to \psi'$ is valid i.e. every state in a coalgebra that satisfies ϕ' also

[1]Strictly speaking they do it for the multisorted version of the syntax but it follows for the single-sorted version

satisfies ψ' and vice-versa. In fact for DA, AGS and Str we have $\phi' = \phi$: no change is needed. The only problem is when K contains $\mathscr{P}_\omega$. Intuitively, to capture relations up to bisimulation one needs Hennessy-Milner logic i.e. conjunctions involving $\Box$s as well as $\Diamond$s. One can overcome this problem by changing the syntax slightly and using $\nabla\{\phi_1,\dots,\phi_n\}$, rather than conjunction and $\Diamond$ in $\mathscr{L}_{\mathscr{P}_\omega:n}$. Modulo these changes:

Theorem 3. *For each $\phi,\psi \in \mathsf{Expr}_K$ one has $\phi \equiv_K \psi$ if and only if $\phi \leftrightarrow \psi$ is valid*

Proof. Just as one associates a regular language to a regular expression one can also associate a set of formulae $Conj_K(\phi) \subseteq \mathscr{L}_{Tree_K}$, to each expression ϕ. Then one shows $\phi \equiv_K \psi \iff Conj_K(\phi) = Conj_K(\psi)$ and that for all ϕ a state in a K-coalgebra satisfies ϕ iff it satisfies every $\chi \in Conj_K$. Briefly, $Conj_{\mathsf{DA}}$, $Conj_{\mathsf{AGS}}$, $Conj_{\mathsf{LTS}}$ assign expressions regular languages, regular languages of guarded strings and formulae of Hennessy-Milner logic, respectively. Moreover the join-semilattice with bottom structure of K – which lifts to the final K-coalgebra – is crucial to the proof. $\Box$

Corollary 1 ([6]). *Decidability of behavioural equivalence follows from the decidability of validity of the coalgebraic μ-calculus.*

Example 4. *The regular expressions for $a \in A$ are defined:*

$$RegExp \ni r ::= \emptyset \mid \varepsilon \mid a \mid r_1 + r_2 \mid r_1 r_2 \mid r^*$$

We define the translation $\tau : RegExp \to \mathsf{Expr}^1_{\mathsf{DA}}$ by: $\tau(\emptyset) = \mathbf{0}$, $\tau(\varepsilon) = \mathbf{1}$, $\tau(a) = \mathbf{a}.\mathbf{1}$, $\tau(r_1 + r_2) = \tau(r_1) \wedge \tau(r_2)$, $\tau(r_1 r_2) = \tau(r_1)[\mathbf{1} := \tau(r_2)]$ and finally $\tau(r^*) = \nu x.(\mathbf{1} \wedge \tau(r)[\mathbf{1} := x])$. Then $\mathsf{Aut}_{\tau(r)}$ is the automaton which accepts r's regular language. Moreover any state in a finite DA-coalgebra which satisfies $\tau(r)$ also accepts each word from this language, although it may accept additional words too. As an exercise one can check that a state satisfies $\tau(aa^*) = \mathbf{a}.\nu x.(\mathbf{1} \wedge \mathbf{a}.x)$ iff it satisfies $\tau(a^*a) = \nu x.(\mathbf{a}.\mathbf{1} \wedge \mathbf{a}.x)$.

The translated formula $\tau(r) \in \mathsf{Expr}^1_{\mathsf{DA}}$ can be exponentially larger than r due to compositions e.g. $\tau((a+b)^n)$ is a binary tree of depth n. However one may coinductively define composition as follows. Let $comp : \mathsf{Expr}_K \times \mathsf{Expr}_K \to \mathsf{Expr}_K$ be $comp(\nu x.\phi, \psi) = comp(\phi[x := \nu x.\phi], \psi)$, $comp(\phi_1 \wedge \phi_2, \psi) = comp(\phi_1) \wedge comp(\phi_2, \psi)$, $comp(\mathbf{a}.\phi, \psi) = \mathbf{a}.comp(\phi, \psi)$, $comp(\top, \psi) = comp(\mathbf{0}, \psi) = \top$ and finally $comp(\mathbf{1}, \psi) = \psi$. Then $\tau((a+b)^n)$ can be represented as $comp(\tau(a+b), comp(\tau(a+b), \dots)$, which avoids the exponential blow up.

References

[1] M. M. Bonsangue, J. J. M. M. Rutten, and A. Silva. Algebras for Kripke polynomial coalgebras. In *Proc. of 24th Ann. IEEE Symp. on Logic in Comput. Sci., LICS 2009 (Los Angeles, Aug. 2009)*, IEEE CS Press, to appear.

[2] M. M. Bonsangue, J. J. M. M. Rutten, and A. Silva. A Kleene theorem for polynomial coalgebras. In L. de Alfaro, ed., *Proc. of 12th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2009 (York, March 2009)*, v. 5504 of *Lect. Notes in Comput. Sci.*, pp. 122–136. Springer, 2009.

[3] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[4] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inform. and Comput.*, 110(2):366–390, 1994.

[5] D. Kozen. On the coalgebraic theory of Kleene algebra with tests. Technical report, Dept. of Computer Sci., Cornell University, 2008. `http://hdl.handle.net/1813/10173`

[6] C. Kupke, C. Cirstea, and D. Pattinson. Complexity of the coalgebraic mu-calculus. In *Proc. of 23rd Int. Wksh. on Computer Science Logic, CSL 2009 (Coimbra, Sept. 2009)*, *Lect. Notes in Comput. Sci.*, Springer, to appear.

[7] J. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[8] L. Schröder and D. Pattinson. Modular algorithms for heterogeneous modal logics. In L. Arge et al., eds., *Proc. of 34th Int. Coll. on Automata, Languages and Programming, ICALP 2007 (Wrocław, July 2007)*, v. 4596 of *Lect. Notes in Comput. Sci.*, pp. 459–471. Springer, 2007.

[9] L. Schröder and D. Pattinson. PSPACE bounds for rank-1 modal logics. *ACM Trans. on Comput. Log.*, 10(2), article 13, 2009.