A description based on languages of the final non-deterministic automaton

A. Ballester-Bolinches a, E. Cosme-Llópeza, R. Esteban-Romero a,b,*

a Departament d’Àlgebra, Universitat de València, Dr. Moliner, 50, E-46100 Burjassot (València), Spain
b Institut Universitari de Matemàtica Pura i Aplicada, Universitat Politècnica de València, Camí de Vera, s/n, E-46022 València, Spain

ARTICLE INFO

Article history:
Received 20 December 2012
Received in revised form 12 December 2013
Accepted 14 January 2014
Communicated by R. van Glabbeek

Keywords:
Non-deterministic automaton
Formal language
Coalgebra
Bisimilarity
Final automaton

ABSTRACT

The study of the behaviour of non-deterministic automata has traditionally focused on the languages which can be associated to the different states. Under this interpretation, the different branches that can be taken at every step are ignored. However, we can also take into account the different decisions which can be made at every state, that is, the branches that can be taken, and these decisions might change the possible future behaviour. In this case, the behaviour of the automata can be described with the help of the concept of bisimilarity. This is the kind of description that is usually obtained when the automata are regarded as labelled transition systems or coalgebras.

Contrarily to what happens with deterministic automata, it is not possible to describe the behaviour up to bisimilarity of states of a non-deterministic automaton by considering just the languages associated to them. In this paper we present a description of a final object for the category of non-deterministic automata, regarded as labelled transition systems, with the help of some structures defined in terms of languages. As a consequence, we obtain a characterisation of bisimilarity of states of automata in terms of languages and a method to minimise non-deterministic automata with respect to bisimilarity of states. This confirms that languages can be considered as the natural objects to describe the behaviour of automata.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The aim of this paper is to present a description of the final object of the category of non-deterministic automata, regarded as labelled transition systems, by means of languages. Our description emphasises the role of languages as natural objects to describe the behaviour of automata.

In this paper we will use the terminology of category theory. We will assume the reader to be familiar with the basic concepts of category theory, as categories, functors, and final or terminal objects. The reader is referred to [20] for more information about category theory.

We can assign to every state of an automaton an associated language, consisting of all words which send this state to a final or terminal state. Traditionally, many authors have considered as the behaviour of a state of an automaton simply its associated language. Under this point of view, the different decisions that may be taken from each state are ignored. However, we can take into account the different branches or decisions that may be taken at every state. They might change the future behaviour of the automaton. From this point of view, automata are regarded as labelled transition systems or...
coalgebras for suitable endofunctors on the category **Set**. In this scope, the idea of the behaviour of the states of the coalgebra is related to the notion of bisimilarity, a concept originated in the field of concurrency (its precise definition will be given in Section 2, see Definition 2.13). We can say that two states have the same behaviour when they are bisimilar. Under very general hypotheses, which hold for automata, when a category of coalgebras possesses a final object, two states are bisimilar if and only if both states have the same image by the unique homomorphism into the final object. This motivates the interest in studying the final objects in some categories of coalgebras, like automata.

Up to now, most known descriptions of final coalgebras are of a very general theoretical nature or are given as a quotient of a coalgebra by the bisimilarity relation. We will present some of them in Section 3. When they are applied to the functor \( N = 2 \times \mathcal{P}_\omega(\text{Id}^A) \) associated to non-deterministic automata, it seems that they do not give a clear idea of the role of languages, which are incontestably a central notion in this theory, in the final automaton. Hence the question of whether languages can be used to describe the behaviour of non-deterministic automata as labelled transition systems remains open. The aim of this paper is to give a positive answer to this question. This also allows us to characterise bisimilarity of states of automata in terms of languages, which has been a long-standing unsolved problem in this theory.

We have done our best to keep our paper self-contained. Accordingly, Section 2 covers several topics of formal languages, automata, and coalgebras. Our main result is presented in Section 3. We conclude the paper by justifying why our description is the most natural one and by establishing some questions for future research.

### 2. Automata and formal languages

An introduction to the classical theory of finite automata can be found in [15]. Since our treatment of automata differs from the usual with respect to the initial state, we have preferred to recall first some basic concepts:

**Definition 2.1.** An alphabet is a finite non-empty set, whose elements are called letters.

**Definition 2.2.** A finite word over an alphabet \( A \) is either the empty word \( \varepsilon \) or a sequence \( a_1a_2\ldots a_n \) of letters of \( A \). The set of all finite words over \( A \) is denoted by \( A^* \).

Note that \( A^* \) can be regarded as the free monoid on the set \( A \), where the multiplication in \( A^* \) is defined as the juxtaposition of words. In the rest of the paper, we will only consider finite words.

**Definition 2.3.** A language (or formal language) over an alphabet \( A \) is a subset of \( A^* \), that is, a set of words over \( A \).

**Definition 2.4** (Operations with languages). If \( L_1 \), \( L_1 \), and \( L_2 \) are languages, we define:

1. the sum \( L_1 + L_2 = L_1 \cup L_2 \) of \( L_1 \) and \( L_2 \), which coincides with the set-theoretical union of \( L_1 \) and \( L_2 \).
2. the product \( L_1L_2 = \{ w_1w_2 \mid w_1 \in L_1, w_2 \in L_2 \} \) of \( L_1 \) and \( L_2 \), composed by the words which are the result of concatenating one word of \( L_1 \) and one word of \( L_2 \), and
3. the Kleene star \( L^* = \bigcup_{n \geq 0} L^n \) of \( L \), where \( L^0 = \{ \varepsilon \} \), \( L^1 = L \) and \( L^{n+1} = L^nL \) for \( n \in \mathbb{N} \).

**Definition 2.5.** The set of all regular languages \( \mathcal{R} \) is the smallest set of languages containing all finite languages and which is closed under taking sums, products, and Kleene stars.

It is usual to identify a letter \( a \) with the language \( \{a\} \). With this criterion, we can identify the regular languages with the so-called regular expressions.

Regular languages are closely connected with finite automata. In this paper we will deal with the next generalisations of the notion of finite automata, in which infinite sets of states are allowed.

**Definition 2.6.** A non-deterministic automaton (respectively, a deterministic automaton, a partial deterministic automaton) is a quadruple \( \mathcal{A} = (S, \Sigma, \delta, \tau) \) in which \( S \) is a set (not necessarily finite) whose elements are called states, \( \Sigma \) is an alphabet, \( \delta \) is a subset of \( S \) whose members will be called final states or accepting states, and the function \( \delta : S \times \Sigma \rightarrow \mathcal{P}(S) \) (respectively, the function \( \delta : S \times \Sigma \rightarrow S \) or the partial function \( \delta : S \times A \rightarrow S \)), called the transition function, assigns to each letter and to each state a finite set of states (respectively, a state, at most one state). When the set of states is finite we say that the corresponding automaton is finite.

Here \( \mathcal{P}(S) \) denotes the set of all finite subsets of the set \( S \). The finiteness restriction on the set of possible transitions from a given state is imposed here to ensure the existence of a final automaton.

It is also common to consider an initial state or a set of initial states in the study of finite automata, but we will not need it in our development, because eventually all states might play the role of the initial state. A deterministic automaton can be considered as a non-deterministic automaton by identifying an image \( s' \) of a state under the transition function with
the singleton \( \{s'\} \). Hence, unless otherwise stated, the word automaton will be used as a synonymous of non-deterministic automaton. We will represent with an arrow \( s_1 \xrightarrow{a} s_2 \) the fact that \( s_2 \in \delta(s_1, a) \).

The transition function of an automaton can be extended to a function \( \delta : S \times A^* \longrightarrow P_\omega(S) \) in the usual way: \( \delta(s, \varepsilon) = \{s\} \), where \( \varepsilon \) is the empty word; if \( w \in A^* \) and \( a \in A \), \( \delta(s, wa) = \bigcup \{\delta(t, a) \mid t \in \delta(s, w)\} \). In the case of deterministic automata, the value of this function is always a singleton. We generalise the notion of language accepted by an automaton with an initial state in the following way:

**Definition 2.7.** Given an automaton \( A = (S, A, S_f, \delta) \) and a state \( s \in S \), the set \( L_{A,s} = \{ w \in A^* \mid \delta(w, s) \cap S_f \neq \emptyset \} \) is called the language accepted or recognised by the automaton \( A \) starting from the state \( s \).

We will write \( L_f \) instead of \( L_{A,s} \) if \( A \) is understood. It is well-known that the regular languages coincide with the languages recognised by finite automata (either non-deterministic, deterministic, or partial deterministic).

All these types of automata and other labelled transition systems can be considered as particular cases of a more general structure, \( \mathcal{F} \)-coalgebras, where \( \mathcal{F} \) is an endofunctor of a category. We will recall here the basic concepts of coalgebras. For a more detailed introduction to the theory of coalgebras, the reader is referred to the works of Adámek [2] or Rutten [21,22]. We will only consider endofunctors of the category Set of all sets and functions between sets.

**Definition 2.8.** Let \( \mathcal{F} \) be an endofunctor of the category Set of all sets and functions. An \( \mathcal{F} \)-coalgebra or \( \mathcal{F} \)-system is a pair \( (S, \alpha_S) \) consisting of a set \( S \) and a function \( \alpha_S : S \longrightarrow \mathcal{F} S \). The set \( S \) is called the carrier of the coalgebra and its elements are called states. The function \( \alpha_S \) receives the name of \( \mathcal{F} \)-transition structure of the system. When \( \alpha_S \) is understood, we use \( S \) instead of \( (S, \alpha_S) \).

In the following, we will denote by \( 2 = \{0, 1\} \) a set of two elements and by \( 1 = \{\ast\} \) a singleton. The identity functor will be denoted by \( \text{id} \).

**Examples 2.9.** (See also [27].) A deterministic automaton \( (S, A, S_f, \delta) \) can be regarded as a \( D \)-coalgebra for the functor \( D = 2 \times \text{id}^A \). Here

\[
\alpha_S(s) = (\alpha_S(s), f_S(s)),
\]

where \( \alpha_S(s) = 1 \) if \( s \in S_f \), \( \alpha_S(s) = 0 \) if \( s \notin S_f \), and \( f_S(s) : A \longrightarrow S \) is defined by \( f_S(s)(a) = \delta(s, a) \). In a similar way, a partial deterministic automaton \( (S, A, S_f, \delta) \) can be regarded as a \( G \)-coalgebra for the functor \( G = 2 \times (1 + \text{id})^A \), where \( \alpha_S(s) = (\alpha_S(s), f_S(s)) \), with \( \alpha_S(s) = 1 \) if \( s \in S_f \), \( \alpha_S(s) = 0 \) if \( s \notin S_f \), and \( f_S(s) : A \longrightarrow 1 + S \) is defined by \( f_S(s)(a) = \delta(s, a) \) if \( \delta(s, a) \) is defined and \( f_S(s)(a) = \ast \) otherwise. Note that the functor used here differs from the one used in [27], which is \( \mathcal{F} = (1 + \text{id})^A \), because we are using a slightly different definition of partial deterministic automata: theirs do not have accepting states. However, our functor coincides with the functor used in [26]. Finally, a non-deterministic automaton \( (S, A, S_f, \delta) \) is an \( N \)-coalgebra for the functor \( N = 2 \times P_\omega(\text{id})^A \). The transition structure \( \alpha_S \) is given by \( \alpha_S(s) = (\alpha_S(s), f_S(s)) \), with \( \alpha_S(s) = 1 \) if \( s \in S_f \), \( \alpha_S(s) = 0 \) if \( s \notin S_f \), and \( f_S(s) : A \longrightarrow P(S) \) given by \( f_S(s)(a) = \delta(s, a) \).

**Definition 2.10.** Let \( (S, \alpha_S) \) and \( (T, \alpha_T) \) be two \( \mathcal{F} \)-coalgebras, where \( \mathcal{F} \) is an endofunctor of Set. A function \( f : S \longrightarrow T \) is a homomorphism of \( \mathcal{F} \)-coalgebras or an \( \mathcal{F} \)-homomorphism if \( (\mathcal{F} f) \circ \alpha_S = \alpha_T \circ f \), in other words, when the following diagram is commutative:

\[
\begin{array}{ccc}
S & \xrightarrow{f} & T \\
\alpha_S \downarrow & & \downarrow \alpha_T \\
\mathcal{F}S & \xrightarrow{\mathcal{F}f} & \mathcal{T}
\end{array}
\]

In order to make the notation lighter, we will follow the standard convention of using the same symbol \( f \) to denote a homomorphism and its underlying function.

We can consider the category of \( \mathcal{F} \)-coalgebras, whose objects are \( \mathcal{F} \)-coalgebras and whose morphisms are \( \mathcal{F} \)-coalgebra homomorphisms.

**Example 2.11.** A homomorphism between the automata \( (S, A, S_f, \delta) \) and \( (S', A, S'_f, \delta') \) is defined by a function \( \phi : S \longrightarrow S' \) such that \( s \in S_f \) if and only if \( \phi(s) \in S'_f \) and \( \delta'(\phi(s), a) = \{\phi(s') \mid s' \in \delta(s, a)\} \). Hence we can consider the category of automata over an alphabet \( A \).

In this paper we will give a description of a final automaton in such a way that the homomorphism from an automaton \( A \) to the final automaton will be given in terms of some structures related to the languages associated with each of the
states of the automaton. These structures will be useful to characterise bisimilarity, a concept originated from concurrency theory in computer science and which is very relevant in the framework of coalgebras.

**Definition 2.12.** Let $\mathcal{F}$ be an endofunctor of $\mathbf{Set}$. Let $(S, \alpha_S)$ and $(T, \alpha_T)$ be two $\mathcal{F}$-coalgebras. A subset $Z \subseteq S \times T$ of the cartesian product of $S$ and $T$ is called an $\mathcal{F}$-bisimulation if there exists a structure function $\gamma : Z \rightarrow \mathcal{F}Z$ such that the projections from $Z$ to $S$ and $T$ are $\mathcal{F}$-coalgebra homomorphisms. In other words, $(Z, \gamma)$ makes the following diagram commute:

\[
\begin{array}{ccc}
S & \xrightarrow{\pi_S} & Z & \xrightarrow{\pi_T} & T \\
\downarrow{\alpha_S} & & \Downarrow{\gamma} & & \downarrow{\alpha_T} \\
F S & \xrightarrow{F \pi_S} & F Z & \xrightarrow{F \pi_T} & FT
\end{array}
\]

If $(S, \alpha_S)$ and $(T, \alpha_T)$ are equal, we simply speak of a bisimulation on $S$.

**Definition 2.13.** Two states $s \in S$ and $t \in T$ are said to be **bisimilar** when there exists a bisimulation $R$ between $S$ and $T$ such that $(s, t) \in R$.

**Example 2.14.** Let $A = (S, A, S_f, \delta)$ and $A' = (T, A, T_f, \delta')$ be two automata over the same alphabet $A$. A relation $R \subseteq S \times T$ is a bisimulation between $A$ and $A'$ if and only if for all $(s, t) \in R$, the following three conditions are satisfied:

1. $s \in S_f$ if and only if $t \in T_f$,
2. for all $s' \in S$, if $s' \in \delta'(s, a)$, then there exists $t' \in T$ such that $t' \in \delta(t, a)$ and $(s', t') \in R$, and
3. for all $t' \in T$, if $t' \in \delta'(t, a)$, then there exists $s' \in S$ such that $s' \in \delta(s, a)$ and $(s', t') \in R$.

Intuitively, we can say that two states of two $\mathcal{F}$-coalgebras are bisimilar when they are not distinguishable from the observer point of view, in other words, when the “observable behaviours” of both automata from both states are the same. This can be used to introduce a semantics in $\mathcal{F}$-coalgebras (see [22,23]). The notion of bisimulation has been studied from a more general point of view in [17] with the help of open maps.

The following technical notion gives a condition which is satisfied by the functors we are interested in. We present it because it is used in the proofs of some of the theorems about bisimulations.

**Definition 2.15.** We say that a weak pullback of two functions $f : X \rightarrow Z$ and $g : Y \rightarrow Z$ in $\mathbf{Set}$ is a triple $(P, \pi_X, \pi_Y)$ such that $P$ is a set, $\pi_X : P \rightarrow X$ and $\pi_Y : P \rightarrow Y$ are functions such that $f \circ \pi_X = g \circ \pi_Y$ and for each triple $(P', \pi'_X, \pi'_Y)$ satisfying the previous conditions, there is a function $p' : P' \rightarrow P$, not necessarily unique, such that $\pi_X \circ p' = \pi'_X$ and $\pi_Y \circ p' = \pi'_Y$. If the function $p'$ is unique, we speak of a pullback. A functor $\mathcal{F} : \mathbf{Set} \rightarrow \mathbf{Set}$ preserves (weak) pullbacks if for every (weak) pullback $(P, \pi_X, \pi_Y)$ of $(f, g)$, the triple $(FP, \pi_X, \pi_Y)$ is a (weak) pullback of $(Ff, Fg)$.

In the following, we will assume that $\mathcal{F}$ is an endofunctor of $\mathbf{Set}$ which preserves weak pullbacks. This assumption holds for the functor $\mathcal{F} = \mathcal{N}$ corresponding to non-deterministic automata, as well as for all other functors presented in this paper (see [22] for more details about this assumption).

The following result summarises some of the properties of bisimulations between $\mathcal{F}$-coalgebras (see [22]). The second statement depends on the fact that $\mathcal{F}$ preserves weak pullbacks.

**Theorem 2.16.** Let $(S, \alpha_S)$ and $(T, \alpha_T)$ be two $\mathcal{F}$-coalgebras.

1. The union of a family of bisimulations between $(S, \alpha_S)$ and $(T, \alpha_T)$ is a bisimulation.
2. The relational composition of two bisimulations between $(S, \alpha_S)$ and $(T, \alpha_T)$ is a bisimulation.
3. The equality relation in $(S, \alpha_S)$ is a bisimulation in $(S, \alpha_S)$.
4. The relational inverse of a bisimulation between $(S, \alpha_S)$ and $(T, \alpha_T)$ is a bisimulation between $(T, \alpha_T)$ and $(S, \alpha_S)$.

As a consequence, there exists a largest bisimulation between two automata over the same alphabet, namely the union of all bisimulations between them.

**Theorem 2.17.** (See [22, Theorem 2.5.]) Let $(S, \alpha_S)$ and $(T, \alpha_T)$ be two $\mathcal{F}$-coalgebras. A function $f : S \rightarrow T$ induces an $\mathcal{F}$-homomorphism between $(S, \alpha_S)$ and $(T, \alpha_T)$ if and only if its graph $G(f) = \{(s, f(s)) \mid s \in S\}$ is a bisimulation between $(S, \alpha_S)$ and $(T, \alpha_T)$.

There has been a big interest in studying the existence and descriptions of final $\mathcal{F}$-coalgebras for a functor $\mathcal{F}$. Let us recall some properties of final coalgebras.
Theorem 2.18. (See [16, Lemma 6.4].) Let $\mathcal{F}$ be an endofunctor on $\text{Set}$.

1. If there exist final $\mathcal{F}$-coalgebras, then all of them are isomorphic.
2. (Lambek’s lemma [19]) If $(T, \alpha_T)$ is a final $\mathcal{F}$-coalgebra, then the function $\alpha_T : T \to \mathcal{F}T$ has an inverse, in other words, $\alpha_T$ is an isomorphism.

The last condition is sometimes expressed in the following terms: a final $\mathcal{F}$-coalgebra $(T, \alpha_T)$ is a fixed point for the functor $\mathcal{F}$.

We cannot ensure the existence of final $\mathcal{F}$-coalgebras for every possible endofunctor of $\text{Set}$. For example, for the functor $\mathcal{F} = \mathcal{P}$ defined by $\mathcal{F}S = \mathcal{P}(S)$, the set of all subsets of $S$, and for a function $f : S \to T$, $\mathcal{F}f(W) = \{f(w) \mid w \in W\}$ for every $W \in \mathcal{P}(S)$, there cannot be any final $\mathcal{P}$-coalgebra: by a well-known theorem of Cantor, the cardinal of $\mathcal{P}(S)$ is strictly smaller than the cardinal of $\mathcal{P}$.

Theorem 2.19 (Rutten and Turi [23]). (See [22, Theorem 9.2].) Every bisimulation of a final $\mathcal{F}$-coalgebra $(T, \alpha_T)$ is contained in the diagonal

$$\Delta_T = \{(t, t) \mid t \in T\}.$$ 

In other words, two bisimilar states are equal.

An $\mathcal{F}$-coalgebra satisfying the above condition (two bisimilar states are equal) is called simple.

A way to check bisimilarity between two states of two $\mathcal{F}$-coalgebras is to check whether both states have the same images under the unique homomorphisms into the final $\mathcal{F}$-coalgebra. This is a consequence of the following result, which is in essence [22, Theorem 4.3] (see also [12, Theorem 5.1, (i) implies (ii)]) and depends on the fact that the functor $\mathcal{F}$ preserves weak pullbacks. We present a proof here for completeness.

Theorem 2.20. Let $(T, \alpha_T)$ be a final $\mathcal{F}$-coalgebra. Two states $s$ and $s'$ of two $\mathcal{F}$-coalgebras $(S, \alpha_S)$ and $(S', \alpha_{S'})$, respectively, are bisimilar if and only if they have the same image under the unique homomorphisms from $(S, \alpha_S)$ and $(S', \alpha_{S'})$ to $(T, \alpha_T)$.

Proof. Denote by $!$ and $'$ the homomorphisms from $S$ and $S'$ to $T$, respectively. Suppose that $!(s) = !(s') = t$, then $(s, t) \in R = G(!)$ and $(s', t) \in R' = G(')$, and $G(!)$ and $G(')$ are bisimulations by Theorem 2.17. Hence $(s, s')$ belongs to the bisimulation $(R')^{-1} \circ R$ by Theorem 2.16 and so $s$ and $s'$ are bisimilar.

Conversely, suppose that $s$ and $s'$ are bisimilar, that is, $(s, s')$ belongs to a bisimulation $V$. Denote by $R = G(!)$ and $R' = G(')$ the graphs of $!$ and $'$, respectively. By Theorem 2.17, $R$ and $R'$ are bisimulations. Then $(s, !(s)) \in R$ and $(s', !(s')) \in R'$. Hence $!(s)$ and $!(s')$ belongs to the bisimulation $R' \circ V \circ R^{-1}$ by Theorem 2.16 and so $!(s)$ and $!(s')$ are bisimilar. Theorem 2.19 shows that $!(s) = !(s')$, as desired. \qed

3. Final automata

In [21], the following description of a final deterministic automaton is presented, which is based on an algorithm of Brzozowski [8]. Let $\mathcal{L} = \mathcal{P}A^*$ be the set of all languages over $A$. Given a word $w \in A^*$, the $w$-derivative or left $w$-quotient of a language $L$ is $L \cdot w = \{v \in A^* \mid vw \in L\}$. A particular case is the $a$-derivative $a^{-1}L = \{v \in A^* \mid av \in L\}$ for $a \in A$, which can be used to give $\mathcal{L}$ a structure of automaton in the following way: $\delta(L, a) = a^{-1}L$ and the language $L$ is final if and only if the empty word belongs to $L$. The language accepted by a state $L$ is precisely $L$ itself. This automaton, called the language automaton, is final and the unique homomorphism from a given automaton $S$ into $\mathcal{L}$ is $!(s) = I_s$, the language accepted by the automaton $S$ when it starts from the state $s$. In particular, two states are bisimilar if and only if the languages accepted by the automaton from these states coincide.

For partial deterministic automata, Silva, Bonchi et al. mention in [26] that the images in the final object for this category of the states of a partial deterministic automaton are pairs of prefix-closed languages $(V, W)$, where $V$ contains all words labelling the paths leading to final states and $W$ contains the words labelling the paths leading to possible states, either final or non-final.

Consider now non-deterministic automata. It is easy to see that bisimilar states accept the same language: Suppose that $s$ and $s'$ are bisimilar states of the automata $(S, A, \delta, S_f)$ and $(S', A, \delta', S'_f)$, respectively, and $w = a_1a_2...a_n \in L_s$, the language associated to $s$ in the first automaton. Then there exists a sequence of states

$$\langle s_0, s_1, s_2, \ldots, s_n \rangle$$

such that $s_0 = s$, $s_i = \delta(s_{i-1}, a_i)$ for $1 \leq i \leq n$, and $s_n \in S_f$. By bisimilarity, there exists a sequence of states $s'_0, s'_1, s'_2, \ldots, s'_n$ such that $s'_0 = s'$ and $s'_i = \delta'(s'_{i-1}, a_i)$ such that $s_i$ is bisimilar to $s'_i$ for $1 \leq i \leq n$. By Example 2.14, then either $s_i$ and $s'_i$ are both final or none of them is final. Since $s_n \in S_f$, it follows that $s'_n \in S'_f$ and so $w \in L'_s$, the language associated to $s$ in the
For all the same extensional tree, and are ing it for a possibly transfinite number of steps. We say that two trees are tree, it is possible to obtain an extensional quotient by identifying two identical subtrees of nodes of the tree and repeat-

ordered finitely branching tree is said to be

is every node is accessible from a fixed node, then it is possible to find an endofunctor $F$ on $\text{Set}$ such that the categories of $F$-coalgebras and $F$-coalgebras are naturally isomorphic.

In their description, they construct a set of expressions based on the elementary components of the functor and an equivalence relation between these expressions. The quotient set of these expressions modulo this equivalence relation admits a structure of a coalgebra for this functor which turns out to be the subcoalgebra of the final coalgebra containing the images of the corresponding finite coalgebras. In their description, they construct a set of expressions based on the elementary components of the functor and an equivalence relation between these expressions. The quotient set of these expressions modulo this equivalence relation admits a structure of a coalgebra for this functor which turns out to be the subcoalgebra of the final coalgebra containing the images of the finite coalgebras.

The finite power-set functor $\mathcal{P}_\omega$ and other related functors on the category $\text{Set}$ have deserved special attention. A non-ordered finitely branching tree is said to be extensional if subtrees rooted at distinct children are not isomorphic. From one tree, it is possible to obtain an extensional quotient by identifying two identical subtrees of nodes of the tree and repeating it for a possibly transfinite number of steps. We say that two trees are extensionally equivalent when they reduce to the same extensional tree, and are similar when the trees of depth $n$ obtained by truncation are extensionally equivalent for all $n$. Barr [5] described the final $\mathcal{P}_\omega$-coalgebra as the quotient coalgebra of the coalegebra composed of all extensional finitely branching trees modulo this relation of similarity. Another relevant description of the final coalegebra for the power-set functor was given by Worrell in [29] (see also Adámek et al. [4]). Let us call a tree $t$ strongly extensional if for every $n$ there exists $m \geq n$ such that the truncation of depth $n$ of $t$ coincides with the truncation of depth $n$ of the result of taking the truncation of depth $m$ of $t$ and collapsing it with respect to extensional equivalence. The set $T$ of all finitely branching, strongly extensional trees has a coalegebra structure $\alpha : T \rightarrow \mathcal{P}_\omega(T)$ assigning to every tree the set of all maximal proper subtrees. This $\mathcal{P}_\omega$-coalgebra is final.

Kozen [18] has presented a combinatorial description of final coalebras on $\text{Set}$. In his work, the role of the functor is played by what he calls a type signature, which is a directed multigraph whose nodes are designated as universal or existential. Universal nodes, denoted by rectangles, correspond to product constructors, while existential nodes, denoted by diamonds, correspond to coproduct constructors. If $F$ is a type signature, an $F$-realisation is a directed multigraph $G$ together with a multigraph homomorphism $l : G \rightarrow F$, called a typing, satisfying the following properties:

- If $l(u)$ is existential, then there is exactly one edge of $G$ with source $u$.
- If $l(u)$ is universal, then $l$ is a bijection between the edges of $G$ with source $u$ and the edges of $F$ with source $l(u)$.

A homomorphism of $F$-realisations is a multigraph homomorphism that commutes with the types.

Let $F$ be a type signature with nodes $V_F$. An $F$-coalegebra is a $V_F$-indexed collection of pairs $(A_s, \alpha_s)$, where the $A_s$ are sets and the $\alpha_s$ are set functions

$$\alpha_s : A_s \rightarrow \begin{cases} \sum_{\text{src} e = s} A_{\text{tgt} e}, & \text{if } s \text{ is existential}, \\ \prod_{\text{src} e = s} A_{\text{tgt} e}, & \text{if } s \text{ is universal}, \end{cases}$$

where src e and tgt e denote, respectively, the source and the target of the arc $e$.

A morphism of $F$-coalebras is a $V_F$-indexed collection of set maps $h_t$ that commute with the $\alpha_s$ in the usual way. This corresponds to the traditional definition of a coalegebra for an endofunctor on $\text{Set}^V$. If the type signature is accessible, that is every node is accessible from a fixed node, then it is possible to find an endofunctor $F$ on $\text{Set}$ such that the categories of $F$-coalebras and $\mathcal{F}$-coalebras are naturally isomorphic.
Kozen showed the existence of a pair of functors between the category of $F$-coalgebras and the category of $F$-realisations, one in each direction, that are inverses up to natural isomorphisms. He proves that these two categories are equivalent and, as a consequence, we can obtain a description of the final $F$-coalgebra from the final $F$-realisation.

The final object for the category of $F$-realisations is showed to be the realisation $\langle R_F, l_F \rangle$ defined as follows. A node of $R_F$ is a set $A$ of finite paths in $F$ such that:

1. $A$ is non-empty and prefix-closed;
2. all paths in $A$ have the same first node, called $l_F(A)$;
3. if $p$ is a path in $A$ of length $n$ and its tail node is existential, then there exists exactly one path of length $n+1$ in $A$ extending $p$;
4. if $p$ is a path in $A$ of length $n$ and its tail node is universal, then all paths of length $n+1$ extending $p$ are in $A$.

The arcs of $R_F$ are defined as follows. Let $A$ be a set of paths in $F$ and $e$ an arc of $F$. The Brzozowski derivative of $A$ with respect to $e$ is the set $D_e(A)$ of paths obtained by removing the initial edge $e$ from all paths in $A$ starting with that edge. If $A$ is a node of $R_F$ and $D_e(A)$ is non-empty, we add exactly one edge $\langle A, e \rangle$ from $A$ to $D_e(A)$ in $R_F$ and we make $l_F(\langle A, e \rangle) = e$. As shown in [18, Theorem 3.2], this realisation is a final object in the category of $F$-realisations.

We have not found in [18] the description of a type signature corresponding to non-deterministic automata. Nevertheless, from the examples in this paper we see that a possible signature type for non-deterministic automata is the graph drawn on Fig. 2, where the nodes with label $t$, $0$, $1$, and $x_i$, $i \in \mathbb{N} \cup \{0\}$, are universal and the node labelled as $2$ and the nodes $w_a$, $a \in A$, are existential; for every $a \in A$ there exists an arc $x_a$ from $t$ to $w_a$ and an arc $v$ from $t$ to $2$; there is an arc $v_0$ from $2$ to $0$ and an arc $v_1$ from $2$ to $1$; from $w_a$ to $x_i$, $i \in \mathbb{N} \cup \{0\}$, there is an arc $x_{ai}$, and from $x_i$ to $t$, $i \in \mathbb{N}$, there are $i$ arcs labelled as $x_{aij}$, $1 \leq j \leq i$.

In the following we will describe an automaton $\mathcal{A}$ as an $F$-realisation $\langle G, l \rangle$. We introduce a procedure to construct a multigraph starting from the graph of the automaton. To every state $s$ in the graph, depending on its nature, we will add the following multigraphs:

- If $s$ is not a final state, we add:

  ![Non-final state multigraph](image)

- If $s$ is a final state, we add:

  ![Final state multigraph](image)

- For every input letter $a$,

  if $s$ has $n$ $a$-labelled outgoing arcs, we replace them by:

  ![Multigraph replacement](image)
if \( s \) has no \( a \)-labelled outgoing arcs, we add:

\[
\begin{align*}
S &\xrightarrow{s_a} s_{a1} \xrightarrow{s_{a2}} s_{a0} \\
\end{align*}
\]

This procedure will give us a multigraph. To complete the description of the realisation we specify its typing \( l \) on the final realisation as follows:

\[
\begin{align*}
\Diamond &\xrightarrow{x_1} \Diamond, \quad \Diamond &\xrightarrow{\Box}, \quad S \xrightarrow{t} t, \\
\end{align*}
\]

**Example 3.2.** Let us exemplify the last procedure on the small automaton \( A = (S, A, S_f, \delta) \) with set of states \( S = \{1, 2, 3\} \), alphabet \( A = \{a\} \), set of final states \( S_f = \{3\} \), and transitions given by \( \delta(1, a) = \{2, 3\} \), \( \delta(2, a) = \{2, 3\} \), \( \delta(3, a) = \emptyset \). This automaton and the result of this procedure are represented in Fig. 3.

The previous description of the final \( F \)-realisation applied to this signature type is the first description we know for the final object for the category of non-deterministic automata that is not given in terms of equivalence classes of a bisimilarity relation, in the sense that in the final automaton, bisimilarity is just a set equality. Kozen also shows at the end of the paper [18] how to characterise the elements of the final realisation as labelled trees.

A slight modification of this type signature, drawn on Fig. 4, gives the type signature corresponding to the \( \mathcal{P}_\omega \)-coalgebras, where \( \mathcal{P}_\omega \) is the finite power-set functor. Its final realisation can be obtained from the strongly extensional trees of Worrell [29] by replacing the edges of the form \( s \rightarrow s' \) by a path \( t \rightarrow x_i \rightarrow t \), where \( i \) is the number of children of \( s \), and a leaf \( s \) is replaced by a path \( t \rightarrow x_0 \). Hence the strongly extensional trees are recovered with this description.

Some recent descriptions of minimisations of non-deterministic automata have been presented by Brzozowski and Tamm [9] and Adámek, Bonchi et al. [3]. We mention them here because they are based on the languages associated to every state of the automaton. However, their way of minimising automata differs from ours, since they only pay attention to the languages associated to every state instead of bisimilarity, as we do. We present them here in order to show the differences with our approach. The problem considered there is the following. Given a regular language \( L \) over an alphabet \( A \) minimal deterministic automata can be considered as canonical acceptors of the given language \( L \). The question is whether it is possible to find an analogous canonical non-deterministic automaton. In [9], the quotients \( L_1, L_2, \ldots, L_n \) of the form \( w^{-1}L \) of a given regular language \( L \) are considered. The non-empty intersections of languages of the form \( L_1 \cap \cdots \cap L_n \) such that \( L_i \) is equal to \( L_j \) or to its complement \( \overline{L_j} \) in which at least one of the \( L_i \) is not complemented are called the atoms of \( L \). The non-deterministic automaton having the atoms of \( L \) as languages as states is called the atomaton of \( L \). For a non-deterministic finite automaton, its determinisation is the deterministic finite automaton obtained by the well-known
subset construction, where only subsets (including the empty subset) reachable from the initial state of \( A \) are used. In [9], the authors show that the determinisation of the átomaton of a regular language \( L \) coincides with the minimal deterministic automaton associated to this language.

In [3], a coalgebraic point of view of this kind of description is presented. However, non-deterministic automata are considered there as coalgebras for the functor \( A \times \text{Id} + 1 : \text{Rel} \to \text{Rel} \), where \( \text{Rel} \) denotes the category of sets and relations. The final object in this category is \( A^* \), and the unique morphism is the relation which assigns to each state all the words sending this state to an accepting state. Under this interpretation, bisimilarity is just language equality. This point of view is different from the one used in this paper. Equivalent descriptions of this automaton can be found in both papers and in the references inside them.

For the case of automata, regarded as labelled transition systems, the previous descriptions do not give, in our opinion, a clear idea of the role of languages in the final automaton. It seems desirable to find, like in the case of deterministic automata, a description which emphasises the role of languages as natural objects to describe the behaviour of automata. This is the aim of the present paper. Our description needs the following concepts.

**Definition 3.3.** A language sequence over an alphabet \( A \) is a finite sequence of the form

\[
\langle L_0, a_1, L_1, a_2, L_2, \ldots, L_{r-1}, a_r, L_r \rangle
\]

where \( L_i \) are languages, that is, elements of \( \mathcal{P}(A^*) \) for \( 0 \leq i \leq r \), \( a_i \in A \) for \( 1 \leq i \leq r \), and \( a_i L_i \subseteq L_{i-1} \) for \( 1 \leq i \leq r \). The number \( r \) is called the length of the language sequence. A sequence formed by a unique language \( L_0 \) will be called a language sequence of length zero.

**Definition 3.4.** A language sequence \( \langle L_0, a_1, L_1, \ldots, L_{r-1}, a_r, L_r \rangle \) over \( A \) is said to be a prefix of the language sequence \( \langle M_0, b_1, M_1, \ldots, M_{s-1}, b_s, M_s \rangle \) over the same alphabet \( A \) when \( r \leq s \) and \( L_j = M_j \) for \( 0 \leq j \leq r \) and \( a_j = b_j \) for \( 1 \leq j \leq r \).

**Definition 3.5.** A language tree is a (possibly empty) set of language sequences \( T \) satisfying the following conditions:

1. Every prefix of a language sequence in \( T \) belongs to \( T \).
2. Given a language sequence

\[
\langle L_0, a_1, L_1, \ldots, L_{k-1}, a_k, L_k \rangle \in T,
\]

the set

\[
N_s = \{ z \in T \mid \text{z is of length } k + 1 \text{ and } s \text{ is a prefix of } z \}
\]

is finite and

\[
L_k \setminus \{ \epsilon \} = \bigcup \{ a_{k+1} L_{k+1} \mid \langle L_0, a_1, L_1, \ldots, L_k, a_{k+1}, L_{k+1} \rangle \in N_s \}.
\]

(1)

When \( N_s = \emptyset \), this union is understood to be \( \emptyset \), and so \( L_k = \{ \epsilon \} \) or \( L_k = \emptyset \).

3. If \( T \) is not empty, then there is a unique language sequence \( \langle L_0 \rangle \) in \( T \) of length zero. The language \( L_0 \) is called the initial language of the language tree.
Definition 3.6. A chain of language trees over an alphabet $A$ is a finite sequence
\[(T_0, a_0, T_1, a_1, T_2, \ldots, T_{r-1}, a_r, T_r)\]
in which $T_i$ is a non-empty language tree over $A$ for $0 \leq i \leq r$, $a_i \in A$ for $1 \leq i \leq r$ such that \{(L_0, a_0, L_1, \ldots, L_r) \mid \langle L_1, \ldots, L_r \rangle \in T_1 \} \subseteq T_0$. The initial language of a chain of language trees $(T_0, a_0, T_1, \ldots, T_r)$ is the initial language of the first language tree $T_0$. The number $r$ is called the length of the chain of language trees. The sequence $T_0$ of a single non-empty language tree over $A$ will be considered a chain of language trees of length zero.

Definition 3.7. A chain of language trees $(T_0, a_1, T_1, \ldots, T_{r-1}, a_r, T_r)$ over $A$ is said to be a prefix of the chain of language trees $(U_0, b_1, U_1, \ldots, U_{s-1}, b_s, U_s)$ over the same alphabet $A$ when $r \leq s$ and $T_j = U_j$ for $0 \leq j \leq r$ and $a_j = b_j$ for $1 \leq j \leq r$.

Now we are in a position to define the states of the final automaton.

Definition 3.8. A tree of chains of language trees over an alphabet $A$ is a set of chains of language trees $T$ satisfying:

1. Every prefix of a chain of language trees in $T$ is also in $T$.
2. Given a chain of language trees
\[U = (T_0, a_1, T_1, \ldots, T_{k-1}, a_k, T_k) \in T,\]
the set
\[N_U = \{V \in T \mid V \text{ is of length } k+1 \text{ and } U \text{ is a prefix of } V\}\]
is finite and
\[T_k = \bigcup \{c(L_k, a_{k+1}, T_{k+1}) \mid (T_0, a_1, T_1, \ldots, T_{k-1}, a_k, T_k) \in N_U\}\]
where $L_k$ is the initial language of $T_k$ and
\[c(L_k, a_{k+1}, T_{k+1}) = \{\langle L_k, a_{k+1}, M_0, b_0, M_1, \ldots, M_r \rangle \mid \langle M_0, b_0, M_1, \ldots, M_r \rangle \in T_{k+1}\} \}

3. There is a unique chain of language trees in $T$ of length zero. Its initial language is called the initial language of $T$ and denoted by $\text{Init}(T)$.

Definition 3.9. The language tree automaton over the alphabet $A$ is
\[L = (S_L, A, S_L, f, \delta_L)\]
such that:

1. $S_L$ is the set of all possible trees of chains of language trees over $A$.
2. a tree of chains of language trees $T$ belongs to the set $S_{L,f}$ of final states if and only if the empty word $\epsilon$ belongs to $\text{Init}(T)$, and
3. given a tree of chains of language trees $T$ and $a_1 \in A$, $\delta_L(T, a_1)$ consists of all trees $U$ of chains of language trees of the form
\[U = \{\langle T_1, a_2, T_2, \ldots, T_{k-1}, a_k, T_k \rangle \mid (T_0, a_1, T_1, a_2, T_2, \ldots, T_{k-1}, a_k, T_k) \in T\},\]
where all chains of language trees of $T$ begin with the language tree $T_0$.

Our next goal is to show that the language tree automaton over the alphabet $A$ is a final object for the category of automata over the alphabet $A$. This will require checking that given an automaton $\mathcal{A}$, there exists a unique homomorphism between $\mathcal{A}$ and $L$. We begin by introducing this homomorphism.

Definition 3.10. Let $\mathcal{A} = (S, A, S_f, \delta)$ be an automaton. Let $q_0 \in S$. A sequence
\[\langle q_0, a_1, q_1, a_2, q_2, \ldots, q_{r-1}, a_r, q_r \rangle\]
with $q_i \in S$, $0 \leq i \leq r$, $a_i \in A$, $1 \leq i \leq r$, and $q_i \in \delta(q_{i-1}, a_i)$ for $1 \leq i \leq r$ will be called a state sequence in $\mathcal{A}$.
Description 3.11. Let $\mathcal{A} = (S, A, S_f, \delta)$ be an automaton. For each state $s \in S$, let $L_s$ denote the language accepted by $\mathcal{A}$ when it starts from $s$. For each state sequence $(q_0, a_1, q_1, a_2, q_2, \ldots, q_r-1, a_r, q_r)$, consider the language sequence $(L_{q_0}, a_1, L_{q_1}, a_2, L_{q_2}, \ldots, L_{q_r-1}, a_r, L_{q_r})$. Let $T_{q_0}$ be the set of all possible sequences which can be obtained in this way from all sequences of states starting with $q_0$. Note that $T_{q_0}$ is a language tree because $L_q \setminus \{\varepsilon\} = \bigcup \{aL_q \mid q' \in \delta(q, a), a \in A\}$ in an automaton for every state $q$. Now for each state sequence $(q_0, a_1, q_1, a_2, q_2, \ldots, q_r-1, a_r, q_r)$ we consider

$$(T_{q_0}, a_1, T_{q_1}, a_2, T_{q_2}, \ldots, T_{q_{r-1}}, a_r, T_{q_r}),$$

which is a chain of language trees. Then the set $Q_{q_0}$ of all chains of language trees which can be obtained from all possible state sequences starting with $q_0$ is a tree of chains of language trees.

Theorem 3.12. Let $\mathcal{A} = (S, A, S_f, \delta)$ be an automaton. The function $\phi : S \rightarrow S_L$ which assigns to each state $s \in S$ the tree of chains of language trees $Q_s$ of Description 3.11 induces a homomorphism of automata between $\mathcal{A}$ and $\mathcal{L}$.

Proof. It is clear that if $s' \in \delta(s, a)$, then $Q_{s'} \in \delta_L(Q_s, a)$. Conversely, suppose that $U \in \delta_L(Q_s, a)$. Then $U$ is a tree of chains of language trees that has been obtained by removing the first element and $a$ in all language sequences in the chains of language trees in $Q_s$ which begin with $(T_q, a)$. But then we get that $U$ is one of the trees of chains of language trees $Q_{s'}$ with $s' \in \delta(s, a)$. Therefore the function $\phi$ respects the transitions. Now assume that $q_0 \in S_f$. Then $\epsilon \in L_{q_0}$. Moreover $\text{Init}(Q_{q_0}) = L_{q_0}$ and since $\epsilon$ is one of the elements of this language, $Q_{q_0} \subseteq S_{L,f}$. On the other hand, if $q_0$ is a final state, then $\epsilon \in \text{Init}(Q_{q_0})$. Hence $\epsilon$ is in the language accepted by $\mathcal{A}$ when it starts from $s$ and so $s \in S_f$. □

Theorem 3.13. Let $\psi$ be a homomorphism between an automaton $\mathcal{A} = (S, A, S_f, \delta)$ and $\mathcal{L}$. Then $\psi$ coincides with the homomorphism $\phi$ of Theorem 3.12. As a consequence, $\mathcal{L}$ is a final object in the category of automata over the alphabet $A$.

Proof. The proof will consist of checking that for every state $q_0 \in S$, $L_{q_0} = \text{Init}(Q_{q_0})$, where $L_{q_0}$ is the language accepted by $\mathcal{A}$ starting from $q_0$ and $Q_{q_0} = \psi(q_0)$. This will be used later to prove that $\psi$ and $\phi$ coincide. For the reader's convenience, we break the proof into stated steps.

1. Let $q_0 \in S$. Then $L_{q_0} \subseteq \text{Init}(Q_{q_0})$. Let $w$ be a word in $L_{q_0}$. If $w = \varepsilon$, then $q_0$ is a final state and so $\psi(q_0)$ is also a final state; in particular, $\varepsilon \in \text{Init}(Q_{q_0})$ where $Q_{q_0} = \psi(q_0)$. Suppose that $w = a_1a_2 \ldots a_r$. Then there exists a state sequence $(q_0, a_1, q_1, a_2, q_2, \ldots, q_{r-1}, a_r, q_r)$ such that $q_r \in S_f$. Then $(Q_{q_0}, a_1, Q_1, a_2, Q_2, \ldots, q_{r-1}, a_r, Q_r)$, where $Q_i = \psi(q_i), 0 \leq i \leq r$, is a state sequence in $\mathcal{L}$ and $Q_r \in S_{L,f}$ is final. Hence $\varepsilon \in \text{Init}(Q_r)$ and so $a_r \in \text{Init}(Q_{r-1}), a_{r-1}a_r \in \text{Init}(Q_{r-2})$, and, by induction, we see that $w = a_1a_2 \ldots a_r \in \text{Init}(Q_{q_0})$. Therefore $L_{q_0} \subseteq \text{Init}(Q_{q_0})$.

2. Conversely, $\text{Init}(Q_{q_0}) \subseteq L_{q_0}$. Consider $w \in \text{Init}(Q_{q_0})$. If $w = \varepsilon$, then $Q_{q_0} \subseteq S_{L,f}$ is final and so $q_0 \in S_f$ is final. Therefore $\varepsilon \in L_{q_0}$. Suppose now that $w = a_1a_2 \ldots a_r$. Note that $Q_{q_0}$ is a tree of language trees and so $Q_{q_0}$ is composed of chains of language trees $(T_0, b_1, T_1, \ldots, T_{r-1}, b_r, T_r)$ satisfying the conditions of Definition 3.8. Now each $T_i$ is a language tree and so it is composed by language sequences $(L_0, c_1, L_1, \ldots, c_{r-1}, L_{r-1})$ satisfying the conditions of Definition 3.6. Let $T_0$ be the unique prefix of length zero of all chains of language trees of $Q_{q_0}$ and let $L_0$ be the unique prefix of length zero of $T_0$. By the condition of Eq. (1) in Definition 3.5, there exists a language $L_1$ such that $a_2 \ldots a_r \in L_1$, and the language sequence $(L_0, a_1, L_1)$ is in $T_0$, there exists a language $L_2$ such that $a_3 \ldots a_r \in L_2$ and $(L_0, a_1, L_1, a_2, L_2) \in T_0$, and, by induction, we see that there exists a language $L_i$ such that the empty word $\varepsilon$ is in $L_i$ and $(L_0, a_1, L_1, a_2, L_2, \ldots, L_{i-1}, a_r, L_r) \in T_i$. By Definition 3.8(2), we obtain that there exists a language tree $T_i$ such that the language sequence $(L_1, a_2, L_2, \ldots, L_{r-1}, a_r, L_r)$ is in $T_i$ and $(T_0, a_1, T_i)$ is a chain of language trees in $Q_{q_0}$, and, once again by induction, we find that there exists a language tree $T_{r-1}$ such that the language sequence $(L_1, a_2, L_2, \ldots, L_{r-1}, a_r, L_r)$ is in $T_{r-1}$ and $(T_0, a_1, T_{r-1})$ is a chain of language trees in $Q_{q_0}$. By Definition 3.9(3), there exists a tree of chains of language trees $Q_i$ such that $(T_1, a_2, T_2, \ldots, T_{r-1}, a_r, T_{r-1})$ is a chain of language trees in $Q_i$ and $(Q_{q_0}, a_1, Q_i)$ is a state sequence in $\mathcal{L}$, and so, with another inductive argument, we find the existence of a tree of chains of language trees $Q_i$ such that $(T_1, a_2, T_2, \ldots, T_{r-1}, a_r, Q_{q_0})$ is a state sequence in $\mathcal{L}$. The state $Q_{q_0}$ is final, because $\varepsilon \in L = \text{Init}(Q_{q_0})$. Since $\psi$ is a homomorphism of automata, there exists a state sequence $(q_0, a_1, q_1, \ldots, q_{r-1}, a_r, q_r)$ in $\mathcal{A}$ such that $\psi(q_i) = Q_i$ for $1 \leq i \leq r$ and $q_r$ is final, because $Q_i$ is final. It follows that $w \in L_{q_0}$. This shows that $\text{Init}(Q_{q_0}) \subseteq L_{q_0}$ for all $q_0 \in S$. 

A. Ballester-Bolinches et al. / Theoretical Computer Science 536 (2014) 1–20
3. The homomorphism $\psi$ coincides with $\phi$.

Now let $\langle q_0, a_1, q_1, \ldots, q_r, a_r, q_r \rangle$ be a state sequence in $A$. Since $\psi$ is a homomorphism of automata, $\langle \psi(q_0), a_1, \psi(q_1), \ldots, \psi(q_r-1), a_r, \psi(q_r) \rangle$ is a state sequence in $L$. By using an argument similar to the one used in the previous paragraph and the fact that the initial language of $\psi(q)$ is $L_q$, we see that the tree of language sequences $T_0$ of the prefix of length zero of $Q_0 = \psi(q_0)$ contains the language sequence $(L_{q_0}, a_1, L_{q_1}, \ldots, L_{q_{r-1}}, a_r, L_{q_r})$.

Now let $\langle L_0, a_1, L_1, \ldots, L_r, a_r, L_r \rangle$ be a language sequence in the tree of language sequences $T_0$ of the prefix of length zero of $\psi(q_0)$. The ideas of the previous paragraph show that there is a chain of trees of language trees $\langle T_0, a_1, T_1, \ldots, T_r, a_r, T_r \rangle$ in which the initial language of $T_i$ is $L_i$ for $0 \leq i \leq r$, and that there exists a state sequence in $L$ of the form $\langle Q_0, a_1, Q_1, \ldots, Q_{r-1}, a_r, Q_r \rangle$ with $\text{Init}(Q_i) = L_i$ for $0 \leq i \leq r$. The fact that $\psi$ is a homomorphism implies that there exists a state sequence

$$\langle q_0, a_1, q_1, \ldots, q_r, a_r, q_r \rangle$$

in $A$ with $Q_i = \psi(q_i)$ and so the language sequence

$$\langle L_0, a_1, L_1, \ldots, L_r, a_r, L_r \rangle$$

coincides with

$$\langle L_{q_0}, a_1, L_{q_1}, \ldots, L_{q_{r-1}}, a_r, L_{q_r} \rangle.$$

It follows that $\psi = \phi$. \qed

Theorems 2.20 and 3.13 yield the following result:

**Corollary 3.14.** Given two automata $(S, A, S_f, \delta)$ and $(S', A, S'_f, \delta')$ over the same alphabet $A$, two states $s \in S$ and $s' \in S'$ are bisimilar if and only if the trees of chains of language trees obtained from $s$ and $s'$ according to Description 3.11 coincide.

**Example 3.15.** Consider the automaton of Example 3.1. We can represent the corresponding trees of language sequences in Fig. 5. Intuitively, what we do to obtain the images in $L$ of each state is to substitute each element of the tree by the complete tree which can be formed from this element. We show it in Fig. 6. The fact that the states 1 and 3 are not bisimilar is shown by the fact that the states 1 and 3 are different (in fact, $T_1$ and $T_3$ are different).

We note that the set of all language sequences obtained from the state 3 is a subset of the set of all language sequences obtained from the state 1. This is the reason we see in $T_1$ only three children $aa^*, a^2a^*$ and $\epsilon$ and in $T_2$ only two children $aa^*$ and $a^2a^*$. However, we cannot determine from the chain of languages $(aa^*, a, aa^*, a, aa^*, \ldots, aa^*, a, aa^*)$ whether it corresponds to the state sequence $(1, a, 1, a, 1, \ldots, 1, a, 1)$, to $(1, a, 1, a, 1, \ldots, 1, a, 3)$, or to $(3, a, 3, a, 3, \ldots, 3, a, 3)$. This distinction appears in the children of the roots of $Q_1$ and $Q_2$.

**Example 3.16.** One might think the final automaton could be described in an easier way by means of the trees of language sequences. In order to show that this is false, we can consider the automaton $A = (S, A, S_f, \delta)$, with $S = \{1, 2, \ldots, 11\}$, $A = \{a, b\}$, $S_f = \{4, 6\}$, $\delta(1, a) = \{8\}$, $\delta(1, b) = \{2, 8\}$, $\delta(2, a) = \{3, 5\}$, $\delta(2, b) = \{5\}$, $\delta(3, a) = \{2\}$, $\delta(3, b) = \{4\}$, $\delta(4, a) = \{4\}$, $\delta(4, b) = \emptyset$, $\delta(5, a) = \{2\}$, $\delta(5, b) = \{4, 6\}$, $\delta(6, a) = \{6\}$, $\delta(6, b) = \emptyset$, $\delta(7, a) = \delta(7, b) = \{8\}$, $\delta(8, a) = \{4, 8\}$, $\delta(8, b) = \{6, 8, 9\}$.
\[ \delta(9, a) = [10, 11], \delta(9, b) = \emptyset, \delta(10, a) = [10], \delta(10, b) = \delta(11, a) = \delta(11, b) = \emptyset. \]

A graphical representation of this automaton appears in Fig. 7.

We can use the Automata package [10] of the computer algebra system GAP [11] to check that the languages associated with each of the states are

\[
\begin{align*}
L_1 &= L_7 = (a + b)^2(a + b)^*, \\
L_2 &= ((a + b)a)^* (a + b)ba^*, \\
L_3 &= L_5 = (a(a + b))^* ba^*, \\
L_4 &= L_6 = a^*, \\
L_8 &= (a + b)(a + b)^*, \\
L_9 &= L_{10} = L_{11} = \emptyset.
\end{align*}
\]

The trees of language sequences corresponding to each state are represented in Fig. 8 and Fig. 9. A branch labelled with more than one letter like \( L_2 \xrightarrow{a,b} L_3 \) is abbreviation of the multiple branch.
Although the language sequences reachable from the states 9 and 10 are the same, these two states cannot be bisimilar, because from 9 we can make a transition with $a$ to the state 11, which has no transitions, but from 10, the only state we can reach is 10, which has a transition labelled with $a$ to this state. This distinction appears when we consider the trees of chains of language trees, which appear in Figs. 10, 11, and 12.

The image of $\mathcal{A}$ in the final automaton is represented in Fig. 13 (the states which are not image of any state of $\mathcal{A}$ are not shown). Note that the only final state is $Q_4$, because the only language containing $\epsilon$ was $L_4$, the initial language of $Q_4$.
Fig. 9. Language sequences of Example 3.16 (continued).

Of course, this also follows from the fact that the final states of $A$, 4 and 6, are mapped into $Q_4$. The automaton shown in Fig. 13 can be regarded as the smallest simple automaton showing the same state behaviour as $A$.

Of course, the subautomaton of $A$ composed by the states 9, 10 and 11 and the corresponding transitions is enough to show that the trees of language sequences are not enough to describe the final automaton. We have presented this more complicated example to show how to work with alphabets consisting of more than one letter.

Example 3.17. Consider now the automaton given by $S = \{1, 2, 3, 4\}$, $A = \{a\}$, $\delta(1) = \{1, 2, 3, 4\}$, $\delta(2) = \{1, 2, 3\}$, $\delta(3) = \{3, 4\}$, $\delta(4) = \emptyset$, and $S_f = \emptyset$. This automaton is like the one in Example 3.1, but with no final states. Obviously, all states have associated the empty language $\emptyset$. The trees of languages associated to this automaton are like the ones represented in
Fig. 10. Trees of chains of trees of languages of Example 3.16.

This technique of considering non-deterministic automata for a language of one letter and no final states can be used to simulate coalgebras for the finite power-set functor $\mathcal{P}_\omega$. Since all languages are empty, the languages turn out to be irrelevant in our discussion for this kind of automata. This can be compared with the description of infinite trees modulo bisimilarity presented by Barr in [5] or the strongly extensional trees of Worrell in [29], which are recovered with our description.

**Remark 3.18.** As we have mentioned in Section 3, automata can be regarded as $F$-realisations in the sense of Kozen [18] for the type signature $F$ of Fig. 2. We now outline how to pass from Kozen’s description to our description and vice versa. The nodes of the final $F$-realisation can be regarded as trees like in the example of Fig. 14, which corresponds to the image in Fig. 5, but with all languages replaced by $\emptyset$. In this case, only the branching information of the automaton is used. The corresponding images in the final automaton look like the ones represented in Fig. 6 with the trees $T_i$ coming from the ones of Fig. 5. The automaton is also simple.
Fig. 11. Trees of chains of trees of languages of Example 3.16 (continued).

the final automaton for the alphabet $A = \{a, b, c\}$ of a final state with two transitions labelled by $a$, a transition labelled by $b$ and no transitions labelled by $c$. We can associate to this state the language corresponding to all words $a_1 \ldots a_k$ such that there exists a path starting with $t$ whose edges are labelled $\langle x_{a_1}, x_{a_1,1}, x_{a_1,1,1}, \ldots, x_{a_k,1}, x_{a_k,1,1}, v, v_1 \rangle$.

We can generate the corresponding language tree by replacing each $t$ by the corresponding language, the path composed by three edges $\langle x_{a_i}, x_{a_i,1}, x_{a_i,1,1} \rangle$ by $a$ and by deleting the paths composed by the edges $\langle v, v_0 \rangle$ or $\langle v, v_1 \rangle$, as well as the paths composed by the edges $\langle x_{a_0} \rangle$. By substituting each state $t$ by the corresponding language tree, we get the tree of chains of language trees of our construction. Conversely, given a state of the language tree automaton and a letter $a \in A$, we can associate to it the following paths:

- First, the path composed by $\langle v, v_1 \rangle$ if $e$ belongs to its initial language and $\langle v, v_0 \rangle$ otherwise.
- Let $a \in A$.
  - If there are no transitions labelled with $a$ from this state, we simply add the path $\langle x_a, x_{a,0} \rangle$.
  - If there are $i$ transitions labelled with $a$ from this state, we assign the paths whose edges have the labels $\langle x_{a_i}, x_{a_i,1}, x_{a_i,1,1} \rangle$, for $1 \leq j \leq i$, followed by all paths corresponding to the images of the transition of this state by $a$ obtained with this method.

The coinduction principle (see Rutten [22]) guarantees that this construction is possible.

4. Discussion and future work

We have obtained a description for the final object in the category of non-deterministic automata in terms of languages. We have also proved that the observational behaviour of an automaton (bisimilarity) can be described in terms of the languages accepted from each state. In our approach, it is just an equality of sets obtained from the languages associated with the states of the automaton. This generalises a known fact for deterministic automata, as the language automaton
introduced in the beginning of Section 3 shows, but which did not seem evident for non-deterministic automata as we have seen in Example 3.1. Our structures derive from the branching information of the automata, with the states substituted by their corresponding languages and, in some sense, follow the same ideas of Barr [5] and Worrell [29] about the branching information. However, even some natural candidates for the states of the final non-deterministic automaton, as the one presented in Example 3.15, based only on the branching information of the automata with the states replaced by their corresponding languages, are not good enough for our purposes.
As we have mentioned, Barr [5] and Worrell [29] have presented a description of final objects in $P_\omega$-coalgebras by means of suitable infinite trees modulo bisimilarity, which exploit their branching information. However, if we want to describe bisimilarity by means of the final object, this approach is not sufficient, because it could be like a petitio principii. A precise description of the relation is achieved in this paper by means of the description of the language tree automaton and the homomorphism from a given automaton to the language tree automaton. Nevertheless, as we have mentioned in the previous section, we obtain trees isomorphic to the strongly extensional trees of Worrell [29] when we use automata to simulate $P_\omega$-coalgebras.

The description of the language tree automaton is indeed a generalisation of the description of the language automaton. In the case of a deterministic automaton, for each state $s$ and each letter $a \in A$, there exists a unique transition $s \xrightarrow{a} t$ and the corresponding languages satisfy the relation $L_t = a^{-1}L_s$. This property also holds in the language automaton, in which the transitions have the form $L \xrightarrow{a} a^{-1}L$. This implies that the language sequences associated to state sequences in a deterministic automaton are uniquely determined by their initial languages. The same can be affirmed about language trees, chains of language trees, and trees of chains of language trees associated to state sequences of deterministic automata, which are also uniquely determined by their initial languages.

The computation of the image of a non-deterministic automaton in the language tree automaton solves a problem of minimisation of automata by Corollary 3.14. The image of a given automaton is a simple automaton, that is, given two different states, they are not bisimilar. The corresponding minimisation problem for deterministic automata is solved by means of the equality of the languages recognised from the states. Other known algorithms are available to identify bisimilarity and so to construct this image into the final automaton, like state partition algorithms (see, for instance, [1]).

We must observe that our automata are not necessarily finite. In fact, the final automaton is infinite. The same thing happens with the final deterministic automaton, whose states are all languages: it is infinite and non-regular languages can appear as states. However, the set of all states reachable from one state by the action of one letter is kept finite in order to make sure that the states of the final object form a set.

A future research line in this subject could be to apply these techniques to study final coalgebras for other structures which can have languages associated with the states in a natural way. This could be an alternative approach to the descriptions of [5–7,18,25,27,29]. For instance, the ideas of Example 3.17 show a possible way to construct the final object for the category of all coalgebras associated with the finite power-set functor.

Another possible future research line could be finding alternative semantics for other structures admitting a coalgebra structure. As an example of what we mean, we might consider the Hennessy–Milner logic. Let $\mathcal{A} = (S, A, S_f, \delta)$ be a non-deterministic automaton. We can define a multi-modal logic $M = M_\mathcal{A}$ with an atomic proposition $p$ whose semantics is given by set of formulas $\mathcal{L}$ defined by the grammar

$$\phi ::= tt \mid \neg \phi \mid \phi_1 \land \phi_2 \mid \Diamond_a \phi \mid p$$

where $a$ varies over $A$. This logic is called Hennessy–Milner logic because it was introduced by Hennessy and Milner in [13, 14] (see also [28] for more details). The usual interpretation of the formulas is given by the modelling relation $\models \subseteq S \times \mathcal{L}$ defined by

- $s \models tt$,
- $s \models \phi_1 \land \phi_2$ if and only if $s \models \phi_1$ and $s \models \phi_2$,
- $s \models \neg \phi$ if and only if $\neg(s \models \phi)$.
\[ s \models \phi \] if and only if there exists \( s' \in \delta(s, a) \) such that \( s' \models \phi \).

\[ s \models p \] if and only if \( s \in S_f \).

The extension of the Hennessy–Milner logic with fixed point operators is the modal \( \mu \)-calculus. A detailed study of the Hennessy–Milner logic and the modal \( \mu \)-calculus, as well as bisimilarity and different semantics for them, can be found in [28]. The trees of chains of language trees over \( A \) defined from the underlying automaton \( \mathcal{A} \) could be used to give an alternative semantics for the Hennessy–Milner logic. We defer the details to a future work.

Acknowledgements

This work has been supported by the grant MTM2010-19938-C03-01 from the Ministerio de Ciencia e Innovación (Spanish Government). The first author has been supported by a research project from the National Natural Science Foundation of China (NSFC, No. 11271085). The second author has been supported by the predoctoral grant AP2010-2764 from the Ministerio de Educación (Spanish Government). We also thank Jean-Éric Pin and Jan Rutten for their helpful comments. Finally, we are indebted to the anonymous referees for their careful reading of the paper and for bringing to our attention some references we had missed. Their suggestions have helped us to improve the presentation of the paper.

References


