Backward Deterministic Büchi Automata on Infinite Words

Thomas Wilke

Department of Computer Science, Kiel University, Germany
thomas.wilke@email.uni-kiel.de

Abstract

This paper describes how backward deterministic Büchi automata are defined, what their main features are, and how they can be applied to solve problems in temporal logic.

1998 ACM Subject Classification automata over infinite objects, modal and temporal logics

Keywords and phrases finite automata, infinite words, determinism, backward automata, temporal logic, separated formulas

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2017.6

Category Invited Paper

1 Introduction

In their famous 1959 paper on finite-state automata [14] Rabin and Scott prove that

1. the reverse of a given non-deterministic finite-state automaton recognizes the reverse of the language recognized by the given automaton and
2. every non-deterministic finite-state automaton can be turned into an equivalent deterministic finite-state automaton.

From an automata-theoretic point of view, these results can be interpreted as follows. The class of regular languages can be defined by either one of the following finite-state automaton models: forward deterministic, forward non-deterministic, backward deterministic, backward non-deterministic.

For infinite words, more precisely, for ω-words, the situation is the same as long as powerful acceptance conditions (such as the parity, Rabin, or Muller condition) are used [8, 13, 10]. For the plain Büchi acceptance condition (or even the more flexible generalized Büchi condition), the situation is different: forward non-deterministic, backward deterministic, and backward non-deterministic automata can be used to define the class of regular ω-languages [2, 3], but forward deterministic automata are strictly weaker [7]. So, in some sense, if one wants a "complete" deterministic automaton model with a simple acceptance condition for regular ω-languages, the model of choice is backward deterministic Büchi automata.

This paper explains how backward deterministic Büchi automata are defined, what their main features are, and how they can be applied to solve problems in temporal logic.

2 Backward deterministic Büchi automata

Backward deterministic Büchi automata were introduced by Carton and Michel in [2, 3], where these automata are termed "complete unambiguous Büchi automata" or "CUBA", for short.

37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017).
Editors: Satya Lokam and R. Ramanujam; Article No. 6; pp. 6:1–6:9

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A backward deterministic Büchi automaton is determined by

- a finite set of states S,
- an initial Büchi condition $B \subseteq S$,
- a final condition $I \subseteq S$, and
- a transition relation $\Delta \subseteq S \times A \times S$ which is reverse deterministic in the sense that there is a function $\delta\colon S \leftarrow A \times S$ such that $\Delta = \{(\delta(a,s),a,s) \mid a \in A, s \in S\}$.

In a backward deterministic generalized Büchi automaton the initial condition is $\mathcal{B} \subseteq 2^S$; the elements of $\mathcal{B}$ are called Büchi sets.

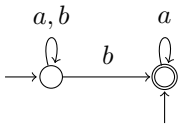
The above only describes the formal ingredients of a backward deterministic (generalized) Büchi automaton. There is also a semantic requirement that needs to be met. It is required that for every ω-word there is exactly one initial run. This means that for every ω-word $u \in A^\omega$ there is exactly one sequence $r \in S^\omega$ such that

- $s_i = \delta(a_i, s_{i+1})$ for every $i \in \omega$ and
- there exist infinitely many i such that $s_i \in B$.

For a generalized Büchi automaton, the second condition is replaced by:

- for every $B \in \mathcal{B}$, there exist infinitely many i such that $s_i \in B$.

Consider the language given by the ω-regular expression $(a+b)^* a^\omega$, which is described by the property "there are only finitely many occurrences of b". There is a simple non-deterministic Büchi automaton for this language:



The reverse of the transition relation is, in fact, a function, but there are no initial runs for the words which belong to the complement of the given language. In other words, this automaton must be extended in order to obtain a backward deterministic Büchi automaton for the language in question. Here is a simple way to do so:

Observe that the second, new component of this automaton has no final states, which is not surprising, because it is just there to make sure that for every word not (!) belonging to the language there is an initial run.

The requirement that there must be exactly one initial run on every word has an immediate important consequence: the automaton obtained from a given backward deterministic (generalized) Büchi automaton by complementing the initial condition (replace I by $S \setminus I$) is a backward deterministic (generalized) Büchi automaton which recognizes the complement of the language recognized by the given automaton:

▶ **Lemma 1** ([3]). *There are simple constructions that*

- *given a backward deterministic Büchi automaton, yield a backward deterministic Büchi automaton recognizing the complement of the language recognized by the given automaton,*
- *given two backward deterministic Büchi automata, yield a backward deterministic Büchi automaton recognizing the intersection (and union) of the language recognized by the given automata.*

The same is true for backward deterministic generalized Büchi automata.

The proof of the second claim is an instance of the question how to turn a backward deterministic generalized Büchi automaton into an equivalent backward deterministic automaton, because by a straightforward product construction one can, given two backward deterministic Büchi automata, construct a backward deterministic generalized Büchi automaton (with two Büchi sets) recognizing the intersection of the languages recognized by the given automata.

▶ **Theorem 2** ([3]). *For every backward deterministic generalized Büchi automaton with n states and m Büchi sets there exists an equivalent backward deterministic Büchi automaton with 2^{mn} states (even if transition conditions are used in the generalized Büchi condition, that is, if $\mathcal{B} \subseteq 2^{S \times A \times S}$).*

To conclude this section, we explain how the semantic property required of a backward deterministic ω-automaton can be checked. Let u be a non-empty finite word, say of length n, and $s \in S$ some state. Let $s_0 = s$ and $s_{i+1} = \delta(u(i), s_i)$ for i with $i < n$. Now u is said to be a loop at s if $s_n = s$ and $\{s_0, \ldots, s_n\}$ satisfies the initial condition, that is, $\{s_0, \ldots, s_n\} \cap B \neq \emptyset$ or $\{s_0, \ldots, s_n\} \cap B \neq \emptyset$ for every $B \in \mathcal{B}$.

▶ **Lemma 3** ([3]). *The semantic requirement (see above) is met if, and only if, every non-empty finite word is a loop at exactly one state. This can be checked in polynomial time.*

3 Completeness

The fundamental theorem on backward deterministic automata is:

▶ **Theorem 4** ([3]). *Every regular ω-language is recognized by a backward deterministic Büchi automaton. More precisely, for every ordinary (forward) non-deterministic Büchi automaton with n states there exists an equivalent backward deterministic Büchi automaton with at most $(12\,n)^n$ states.*

This theorem is due to Carton and Michel, as stated above. In their paper [3], Carton and Michel describe two different proofs. Another proof can be found in the book [11] by Perrin an Pin an yet another proof is given in [19].

The starting point for most of these proofs is an analysis of accepting runs (run DAGs) of ordinary Büchi automata.

4 Application to temporal logic

Backward deterministic ω-automata go very well with temporal logic—this is explained in the rest of this paper.

We consider both future-only linear-time temporal logic and future/past linear-time temporal logic. That is, we consider formulas which are built from propositional variables such as p and q using boolean operators and the temporal operators X (next), F (eventually), G (globally), U (until) for the future-only logic and these operators plus their past counterparts ($\overleftarrow{\mathsf{X}}$, $\overleftarrow{\mathsf{F}}$, $\overleftarrow{\mathsf{G}}$, and $\overleftarrow{\mathsf{U}}$) for the future/past logic.

A typical formula is

$$\mathsf{GF}p \ ,$$

which is to be read "always eventually p" (see above) and which means p holds infinitely often when the domain of time is, for instance, the canonical order of the natural numbers. In fact, in this paper we interpret temporal formulas in finite words or ω-words. The positions

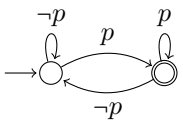
of such a word form the domain of time, the symbol at a position is a set of propositional (boolean) variables, exactly those variables that are true in the position. For instance, we have

$\{p\}\{q\}\{p\}\{q\}\ldots \vDash \mathsf{GF}p$, $\{q\}\{\}\{q\}\{\}\ldots \nvDash \mathsf{GF}p$.

4.1 From formulas to automata and back

There is a fundamental result on temporal logic and finite-state automata which says that a regular language of finite words is expressible in future-only or future/past temporal logic if, and only if, it is recognized by a counter-free automaton [6, 5, 9, 16]. Here, a state s is a mod-n counter for a non-empty finite word u in a given finite-state automaton with a transition function denoted δ if the states $s_0, s_1, \ldots, s_n$ defined by $\delta(s_i, u) = s_{i+1}$ for every $i < n$ are all distinct and $s_0 = s_n$ holds.

For example, consider the formula from above, which on finite words says that p holds in the last position, and a forward deterministic automaton for this language:



There is no counter in this automaton, because every word that leads from the left to the right state ends in $\{p\}$ and every word that leads from the right to the left state ends in $\{\}$.—Observe that we do not really attach symbols to the transitions in the diagram, but propositional formulas. Each such formula represents the variable assignments (which are the real symbols) that satisfy it.

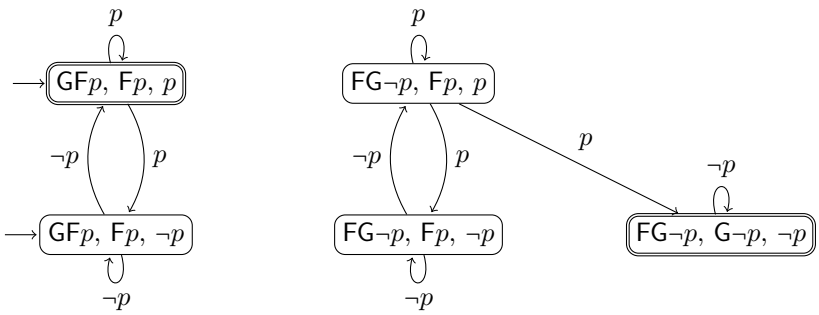
For proving the two directions of the above result it is easier to work with backward deterministic automata than with ordinary (forward) deterministic automata, or, alternatively, with past temporal formulas instead of future temporal formulas: the straightforward translation of future temporal formulas into non-deterministic automata [17], where the automaton guesses for each subformula where it is true in the given word and where it is not true, yields a backward deterministic automaton, even a backward deterministic counter-free automaton. For the other direction, from backward deterministic counter-free automata to future temporal formulas, a nested inductive argument can be used, see [18].

For ω-languages, that is, for future-only temporal logic on ω-words, similar techniques can be applied when working with backward deterministic (generalized) Büchi automata and a corresponding result holds:

▶ **Theorem 5** ([20]). *An ω-language is expressible in future temporal logic if, and only if, it is recognized by a counter-free backward deterministic (generalized) Büchi automaton.*

In the inductive proof of the more difficult direction, from automata to formulas, the corresponding result on finite words is used.

Here is the result of the straightforward translation of the formula $\mathsf{GF}p$ from above:



This automaton is, in fact, a backward deterministic Büchi automaton, and it is counter-free for similar reasons as in the previous example.—Note that the usual construction for turning a formula into an automaton results in a generalized Büchi condition with as many Büchi sets as there are F and U subformulas. So, here, we would expect two such sets, but these two are equivalent to the one given in the diagram.

4.2 Separated formulas

For future/past temporal formulas, there is no syntactic restriction on nesting future and past modalities. For instance,

$$\mathsf{F}(q \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}\,p) \tag{1}$$

is, indeed, a formula. It is to be read "currently or sometime in the future q holds and p holds all the time before".

Here is a small diagram illustrating the property:

$$q\ q\ \ldots\ q\ q\ \boxed{q}\ q\ q\ \ldots\ q\ q\ p\ \ldots,$$

where the box denotes the present.

Obviously, the property can also be phrased "q holds all the time in the past and q holds until p holds sometime in the future or p holds currently". The corresponding formula is

$$\overleftarrow{\mathsf{G}}\,q \wedge q\mathsf{U}p\ . \tag{2}$$

This formula is such that past and future modalities are not nested. It is even true that the formula is a boolean combination of pure past, pure future and present (propositional) formulas.—When this is the case one says that a formula is separated.

A fundamental theorem by Gabbay states that every future/past temporal formula is equivalent to a separated one [4], see also [5]. In the following, it is explained how this result can be proved for the natural numbers using backward deterministic Büchi automata.

Unlike with future-only temporal formulas, which are usually interpreted in the first position of an ω-word, future/past formulas are interpreted in any position of an ω-word, as we have just seen. So the view that such a formula defines a set of ω-words is not appropriate anymore. Rather, it assigns to each position in an ω-word a truth value, namely whether or not the formula is true in that position. Formally, we can think of such a formula to define a function $A^\omega \to \{0,1\}^\omega$, where A is the underlying alphabet, for instance $2^{\{p,q\}}$.

From an automata-theoretic perspective, this requires a different automaton model. In fact, one would like to have an automaton model which describes, for every given ω-word

over the right alphabet, an ω-word over $\{0,1\}$, encoding where the formula in question is true, or, in other words, the value of the above function for this ω-word.

An appropriate automaton model one can work with is that of bimachine [15, 1]. A bimachine is given by

- a forward deterministic semi-automaton (no final state set) on finite words,
- a backward deterministic semi-automaton (no final state set) on ω-words, and
- a function $\lambda\colon Q \times A \times S \to \{0,1\}$.

Such a bimachine defines, indeed, a function $A^\omega \to \{0,1\}^\omega$. We describe how the value for some ω-word u is determined. Let us denote this value by w—recall that w is an ω-word over the alphabet $\{0,1\}$. Let $i \in \omega$ be any position. Then there is a unique state that the forward automaton assumes after having read the prefix $u(0)...u(i-1)$, say q. Similarly, there is a unique state that the backward deterministic semi-automaton assumes after having read the suffix $u(i+1)u(i+2)\ldots$, say s. Now, $w(i) = \lambda(q, u(i), s)$.

In other word, we let run the two automata from the start and the end of the word to the position in question and then combine the resulting states with the symbol of u at that position in order to determine the symbol of w at the same position:

$$q_0u(0)q_1u(1)q_2\ldots u(i-1)q_i \ , \qquad w(i) = \lambda(q_i, u(i), s_0) \ , \qquad s_0u(i+1)s_1u(i+2)\ldots \ .$$

The fundamental theorem now is:

▶ **Theorem 6** ([20]). *A function $A^\omega \to \{0,1\}^\omega$ is expressible in future/past temporal logic if, and only if, it is realized by a counter-free bimachine.*

Here, counter-free means that the forward and the backward automaton are counter-free.

The direction from left to right can be proved by an inductive argument where the inductive step involves an involved automata-theoretic construction.

The other direction follows from the fact that counter-free automata on finite words and backward deterministic counter-free Büchi automata define temporal properties. In fact, the prove yields immediately:

▶ **Corollary 7** ([4, 5]). *Every future/past temporal formula is equivalent to a separated formula.*

4.3 Effective characterizations of temporal logic and its fragments

The crucial ingredient of temporal logic are its temporal operators. Therefore, questions like "Can a given property be expressed

- without X?"
- with F only?"
- without nesting U?"
- by nesting U at most k times?"

are obvious questions to be asked. A similar question is: "Is a given regular ω-language definable in temporal logic at all?"—Backward deterministic ω-automata are a good means for designing corresponding decision procedures.

To describe appropriate decision criteria, we need some more preparation. We need to introduce a congruence relation on the state space of a given backward deterministic (generalized) Büchi automaton and to explain the notion "pattern".

Assume we are given a backward deterministic (generalized) Büchi automaton. Then this automaton has a deterministic transition function, say $\delta\colon S \leftarrow A \times S$ and a final state set $I \subseteq S$. We define an equivalence relation denoted $\equiv$ on S by: $s \equiv s'$ iff for every $u \in A^*$,

$\delta(u,s) \in I$ iff $\delta(u,s') \in I$. In a certain sense, this is a left congruence: whenever $s \equiv s'$, then $\delta(u,s) \equiv \delta(u,s')$ for every $u \in A^*$. Therefore, we can factor the transition function δ through $\equiv$. That is, a function $\delta_{\equiv}\colon S/{\equiv} \leftarrow A \times S/{\equiv}$ can be defined by

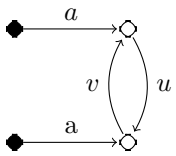
$$\delta_{\equiv}(a, s/{\equiv}) = \delta(a,s)/{\equiv} \quad . \tag{3}$$

An almost immediate consequence of Theorem 5 is the following characterization of temporal logic:

▶ **Theorem 8.** *An ω-language recognized by a backward deterministic (generalized) Büchi automaton with reverse transition function δ is definable in temporal logic if, and only if, the quotient transition function $\delta/{\equiv}$ does not have a counter.*

This answers the last question of the ones posed above. To show how the answer to one of the other question looks like, we consider F expressibility—the second question. We ask whether a given temporal property is expressible in the fragment of future temporal logic where F is the only temporal operator.

To describe an appropriate answer we need the notion "pattern". Here is an example of a pattern:



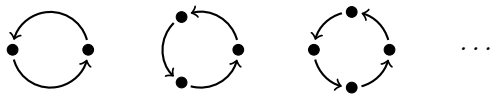
Such a graph is interpreted in the transition graph induced by a transition function δ. Symbols like a and b are variables for elements of the alphabet, symbols like u and v are variables for words over the alphabet, and full circles stand for distinct (!) states. So a backward deterministic automaton matches the above pattern if there are states s_0, s_1, s_2, and s_3 such that the following is true:

- there is a symbol $a \in A$ such that $s_3 = \delta(a, s_2)$ and $s_0 = \delta(a, s_1)$;
- there is some $u \in A^*$ such that $s_2 = \delta(u, s_1)$ and there is some $v \in A^*$ such that $s_1 = \delta(v, s_2)$;
- the states s_0 and s_3 are distinct.

Another way to put this is to say that there are states s_1 and s_2 in the same strongly connected component of the transition graph induced by δ and a symbol $a \in A$ such that $\delta(a, s_1) \neq \delta(a, s_2)$.

So Theorem 8 can be rephrased as follows.

▶ **Theorem 9** (Theorem 8 rephrased)**.** *An ω-language recognized by a backward deterministic (generalized) Büchi automaton with reverse transition function δ is definable in temporal logic if, and only if, the quotient transition function $\delta/{\equiv}$ does not have one of the following patterns.*

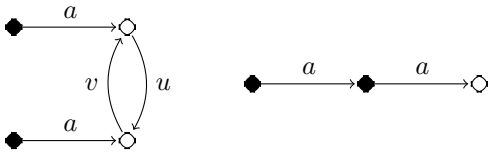


Recall the notion "loop" from above. In order to be able to state the criterion for F expressibility in a concise fashion, we need some more notation. When $u \in A^+$, we write $u\circlearrowright$ for the state at which u is a loop. We also need the notation $\mathrm{occ}(u)$ to denote the set of symbols occurring in a word u.

Now, the criterion for F expressibility is as follows.

▶ **Theorem 10** ([12]). *A temporal property is expressible in the* F *fragment of future temporal logic if, and only if, the following holds for any backward deterministic automaton for the ω-language defined by the property.*

1. *The quotient transition function $\delta/\equiv$ does not have one of the following patterns.*



2. *For all non-empty finite words u and v over the respective alphabet, if $u(0) = v(0)$ and $occ(u) = occ(v)$, then $u\wr = v\wr$.*

Observe that the second pattern states that the property is stutter-invariant: if an ω-word results from another ω-word by compressing or deflating chunks consisting of the same symbol, then either both words are accepted or neither one of them. This is an obvious requirement because the operator F is stutter-invariant.

5 Open problems

For several of the constructions with backward deterministic (generalized) Büchi automata, the exact complexity is not known. For instance, it would be good to have a reasonable upper bound for the translation of a future/past temporal formula into a counter-free bimachine.—Unlike with future-only temporal logic, the automata-theoretic constructions are involved. In fact, one step in the corresponding construction incurs an exponential blowup.

References

1 Olivier Carton. Right-sequential functions on infinite words. In Farid M. Ablayev and Ernst W. Mayr, editors, *Computer Science – Theory and Applications, 5th International Computer Science Symposium in Russia, CSR 2010, Kazan, Russia, June 16-20, 2010. Proceedings*, volume 6072 of *Lecture Notes in Computer Science*, pages 96–106. Springer, 2010. doi:10.1007/978-3-642-13182-0_9.

2 Olivier Carton and Max Michel. Unambiguous büchi automata. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 407–416. Springer, 2000. doi:10.1007/10719839_40.

3 Olivier Carton and Max Michel. Unambiguous büchi automata. *Theor. Comput. Sci.*, 297(1-3):37–81, 2003. doi:10.1016/S0304-3975(02)00618-7.

4 Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In Behnam Banieqbal, Howard Barringer, and Amir Pnueli, editors, *Temporal Logic in Specification, Altrincham, UK, April 8-10, 1987, Proceedings*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer, 1987. doi:10.1007/3-540-51803-7_36.

5 Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In Paul W. Abrahams, Richard J. Lipton, and Stephen R. Bourne, editors, *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980*, pages 163–173. ACM Press, 1980. doi:10.1145/567446.567462.

6 Johan Anthony Willem Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.

7 Lawrence H. Landweber. Decision problems for omega-automata. *Mathematical Systems Theory*, 3(4):376–384, 1969. doi:10.1007/BF01691063.

8 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966. doi:10.1016/S0019-9958(66)80013-X.

9 Robert McNaughton and Seymour Papert. *Counter-free automata*. M.I.T. Press research monographs. M.I.T. Press, 1971. URL: https://books.google.de/books?id=QRbvAAAAMAAJ.

10 Andrzej Wlodzimierz Mostowski. Regular expressions for infinite trees and a standard form of automata. In Andrzej Skowron, editor, *Computation Theory – Fifth Symposium, Zaborów, Poland, December 3-8, 1984, Proceedings*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1984. doi:10.1007/3-540-16066-3_15.

11 Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. Number 141 in Pure and Applied Mathematics. Elsevier, Amsterdam, 2004.

12 Sebastian Preugschat and Thomas Wilke. Effective characterizations of simple fragments of temporal logic using carton–michel automata. *Logical Methods in Computer Science*, 9(2), 2013. doi:10.2168/LMCS-9(2:8)2013.

13 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969. doi:10.2307/1995086.

14 Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959. doi:10.1147/rd.32.0114.

15 Marcel Paul Schützenberger. A remark on finite transducers. *Information and Control*, 4(2-3):185–196, 1961. doi:10.1016/S0019-9958(61)80006-5.

16 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.

17 Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994. doi:10.1006/inco.1994.1092.

18 Thomas Wilke. Classifying discrete temporal properties. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 1999. doi:10.1007/3-540-49116-3_3.

19 Thomas Wilke. ω-automata. *CoRR*, abs/1609.03062, 2016. arXiv:1609.03062.

20 Thomas Wilke. Past, present, and infinite future. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 95:1–95:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.95.