Natural Colors of Infinite Words

Rüdiger Ehlers
Technische Universität Clausthal, Germany

Sven Schewe
University of Liverpool, UK

Abstract

While finite automata have minimal DFAs as a simple and natural normal form, deterministic omega-automata do not currently have anything similar. One reason for this is that a normal form for omega-regular languages has to speak about more than acceptance – for example, to have a normal form for a parity language, it should relate every infinite word to some natural color for this language. This raises the question of whether or not a concept such as a natural color of an infinite word (for a given language) exists, and, if it does, how it relates back to automata.

We define the natural color of a word purely based on an omega-regular language, and show how this natural color can be traced back from any deterministic parity automaton after two cheap and simple automaton transformations. The resulting *streamlined* automaton does not necessarily accept every word with its natural color, but it has a "co-run", which is like a run, but can *once* move to a language equivalent state, whose color is the natural color, and no co-run with a higher color exists.

The streamlined automaton defines, for every color c, a good-for-games co-Büchi automaton that recognizes the words whose natural colors with respect to the represented language are at least c. This provides a canonical representation for every ω-regular language, because good-for-games co-Büchi automata have a canonical minimal – and cheap to obtain – representation for every co-Büchi language.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Linear logic; Theory of computation → Logic and verification

Keywords and phrases parity automata, automata over infinite words, ω-regular languages

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2022.36

Funding *Rüdiger Ehlers*: This work was supported by the German Science Foundation (DFG) under Grant No. 322591867 (GUISynth).
Sven Schewe: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 956123 (FOCETA). It was also supported by the EPSRC through project EP/X017796/1.

Acknowledgements We thank Arved Friedemann for interesting discussions that had a positive influence on the undertaking of this work.

1 Introduction

A classical question in the theory of automata is how to define *canonical representations* of regular languages. Such a representation, typically in the form of an automaton, for a language has several advantages. For once, different canonical automata must define different languages. But reasonably defined canonical automata are also concise and normally a minimal (and thereby natural) representative of all language equivalent automata of the same type, which makes them a natural representative of the language they recognize.

Such definitions of canonicity build on – or deliver – insights into the possible structure of an automaton for a given language. For instance, canonical deterministic automata over finite words have exactly one state per (reachable) suffix language, and the Myhill-Nerode automaton minimization procedure is able to translate every deterministic automaton over finite words into its canonical form in polynomial time [6]. The concepts that underpin insightful canonicity definitions often give rise to efficient minimization procedures, which makes it attractive to apply them in practical applications. In turn, such concepts are useful in concisely defining a language, including for practical applications like learning, model checking, or synthesis.

For regular languages over infinite words, obtaining insightful canonical forms has remained a challenge. Such languages are useful for reasoning about *reactive systems*, i.e., computational systems that continuously read from an input stream while producing an output stream. While Löding [8] gave a construction for computing canonical and minimal deterministic *weak* automata, which can encode some such languages, this automata class is very restricted in that in every strongly connected component, either all states are accepting or all states are rejecting. This means that simple languages such as "there are (in)finitely many as in the word" cannot be represented by them.

After the result by Löding [8], there was, for quite a while, little progress on canonical forms for more expressive subclasses of ω-regular languages. This is partially rooted in the fact that deterministic Büchi (and co-Büchi) automata – which are among the simplest ω-automaton types and cannot even capture all ω-regular languages – have an NP-complete minimization problem [11]. This implies that, unlike in the case of languages over finite words, deterministic automata with the common state based acceptance cannot be used for defining a canonical form that is easy to compute.

Only very recently, Abu Radi and Kupferman [1] observed that, via a slight generalization from deterministic co-Büchi automata to (transition-based) good-for-games [5] co-Büchi automata, we obtain an automaton model for co-Büchi languages that permits a polynomial-time minimization procedure; this gives rise to an insightful canonical form. Transition-based good-for-games co-Büchi automata (and similarly, good-for-games parity automata with a fixed set of colors) are not more expressive than deterministic automata with the same acceptance condition [5], but they can be more concise. Interestingly, this added conciseness is what enables polynomial-time minimization and thereby efficiently computing canonical automata. In the canonical minimal automata computed using the construction by Abu Radi and Kupferman [1], non-determinism only appears along *rejecting* transitions, which connect different strongly connected components that consist only of accepting transitions. Hence, the different deterministic strongly connected components represent the different ways in which a word can be accepted and hence provide insight into the structure of the represented language.

The result by Abu Radi and Kupferman raises the question of whether this result can be extended to obtain a canonical and insightful representation for general ω-regular languages or not. Such an extension would intuitively need to use a richer type of acceptance condition than co-Büchi acceptance, as co-Büchi acceptance is too limited in expressivity. The weakest acceptance condition that offers full ω-regularity in this context is *parity acceptance*. In parity automata, a word is accepted if, and only if, the lowest *color* that occurs infinitely often along a run of the automaton is even.

We could salvage the polynomial-time canonicalization procedure for co-Büchi acceptance while using a parity-type acceptance condition by representing an ω-regular language L by a falling chain of languages $L_0 \supset L_1 \supset L_2 \ldots \supset L_c$ such that L_0 is the universal language and each language L_i is a co-Büchi language. A word is then accepted by the *chain* of languages if, and only if, the highest i such that the word is in L_i is even. As all of these languages are co-Büchi languages, we can represent each of them by their canonical minimal good-for-games automaton such that, together, these automata are a canonical representation of L.

The crucial piece that is currently missing in the literature to obtain such a canonical representation of an arbitrary ω-regular language, however, is which word should be in which language L_i. Omega-regular languages can be decomposed into such chains in different ways, and for the overall chain to be a canonical representation of the language, we need to fix a way for decomposing the language L into co-Büchi languages L_i. In other words, we are missing a definition of the *natural* color of a word that defines the highest index i such that the word is in L_i. This natural color depends on the overall language to be represented, as this color reflects where in the decomposition of a given language the word resides.

For a useful canonicity definition, we need the allocation of words to the individual L_i for a given ω-regular language L to have several properties:

1. the definition should be based on the language L alone, and be independent of the syntactic structure of any representation of it (such as some parity automaton that recognizes L),
2. the definition should be easy to compute for a given word and a given representation of L (such as a deterministic parity automaton), and
3. starting from an automaton representation of L, the sizes of co-Büchi automata for the languages L_i should be *small*, ideally not bigger than the size of an automaton for L.

In this paper, we provide a definition of a *natural* color of an infinite word for a given ω-regular language that has these properties. Our definition distills the idea that, in a parity automaton, only the lowest color visited infinitely often along a run matters, into a concept that can be defined on languages alone, without referring to a specific automaton. We then use this for introducing a canonical representation of arbitrary ω-regular languages as a chain of co-Büchi languages. While the definition of the natural color of a word (for a given language) is the main technical contribution of this paper, its study is motivated by what it can be used for, namely for establishing a canconical representation for ω-regular languages, which is the conceptual contribution of this paper.

We show that our particular definition of the natural color of a word (for a given language) has the property that every deterministic parity automaton can be translated into a form from which the natural color of a word can easily be read off. This works in two steps: We first simplify an automaton by ordering its strongly connected components (using an order that respects reachability) and bending all transitions to language equivalent states in the maximal component they reside in (besides removing unreachable and unproductive states). In a second step, we construct a so-called *streamlined* form of a parity automaton that retains the transition structure. Both transformations are tractable.

From a streamlined automaton, we can furthermore obtain, again in polynomial time, good-for-games co-Büchi automata for all languages L_i. They are no larger than the original streamlined parity automaton, and therefore no larger than the deterministic parity automaton we started with. Moreover, they can subsequently be minimized [1] to obtain a canonical representation of a given ω-regular language. This minimization can also yield an exponential advantage over a representation as deterministic co-Büchi automata [7].

As a consequence, with our definition, one can obtain, in polynomial time, a canonical representation of the language of a deterministic parity automaton. While this representation is not a single automaton, deviating from deterministic branching was necessary in order to avoid the NP-hardness of minimizing deterministic parity automata. Furthermore, it was shown that good-for-games parity (incl. Büchi and co-Büchi) automata are also NP-hard to minimize [12] when using state-based acceptance, while the complexity of minimizing Büchi (and, more generally, parity) good-for-games automata with transition-based acceptance is still open, so a further generalization had to be made. By choosing a sequence of good-for-games transition-based co-Büchi automata as this generalization, we avoid introducing a more complex automaton type at the cost of having multiple automata.

While it is possible to define other variants of what the natural color of a word (for a given language) could be, our definition has the advantage that it coincides with the color of a word of *some* parity automaton for a given language while permitting a translation from a deterministic parity word automaton to a canonical representation of its language in polynomial time.

2 Preliminaries

Given a set S, we denote the set of finite sequences (words) of elements in S as S^* and the set of infinite sequences of elements in S as S^ω. Sets of words are also called *languages* (over some alphabet). We only consider finite alphabets. We denote the set of natural numbers including 0 by $\mathbb{N}$. Given a language $L \subseteq \Sigma^\omega$ and a finite word $w \in \Sigma^*$, we define the suffix language of L over w as $\mathcal{L}^{suffix}(L, w) = \{w' \in \Sigma^\omega \mid ww' \in L\}$.

We define *parity automata (with transition-based acceptance)* as tuples $\mathcal{A} = (Q, \Sigma, \delta, Q_0)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta \subseteq Q \times \Sigma \times Q \times \mathbb{N}$ is a transition relation, and Q_0 is the set of initial states. We say that a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ induces a run $\pi = q_0 q_1 \ldots$ of the automaton with the corresponding color sequence $\rho = \rho_0 \rho_1 \ldots \in \mathbb{N}^*$ if $q_0 \in Q_0$ and, for every $j \in \mathbb{N}$, we have $(q_j, w_j, q_{j+1}, \rho_j) \in \delta$. We say that w is *accepted* by $\mathcal{A}$ if there exists a run ρ for the word on which the lowest color that occurs infinitely often along ρ is even. For the remainder of this paper, all automata considered employ transition-based acceptance whenever not stated otherwise.

We say that $\mathcal{A}$ is deterministic if, for every state $q \in Q$ and $x \in \Sigma$, we have exactly one element of the form $(q, x, q', c) \in \delta$, and Q_0 is a singleton set. We henceforth use q_0 as tuple element for the initial state for deterministic automata. In deterministic automata, every word induces a unique run/color sequence combination. This also allows us to define, by slight abuse of notation, for each state q and word $w \in \Sigma^*$, $\delta(q, w)$ to be the unique state reached from q after reading w. We refer to the smallest color that occurs infinitely often in the color sequence corresponding to a run for w as the *color of w in $\mathcal{A}$*. We also call it the *dominating color* among the ones occurring infinitely often in the color sequence for w.

The set of words accepted by $\mathcal{A}$ is called its language, also denoted by $\mathcal{L}(\mathcal{A})$. We say that $\mathcal{A}$ is a co-Büchi automaton if only the colors 1 and 2 occur along transitions. Transitions with color 1 and 2 are also called *rejecting* and *accepting* transitions, respectively. The automaton $\mathcal{A}_q$ with $q \in Q$ denotes a variant of $\mathcal{A}$ for which the initial state has been replaced by q. We say that two states $q, q' \in Q$ of a deterministic parity automaton (*DPA*) $\mathcal{A}$ are *equivalent*, denoted by $q \sim_\mathcal{A} q'$, if, and only if, they have the same language $\mathcal{L}(\mathcal{A}_q) = \mathcal{L}(\mathcal{A}_{q'})$.

We say that an automaton $\mathcal{A}$ is *good-for-games* if there exists a strategy function [1] $f : \Sigma^* \to Q \times \mathbb{N}$ such that for each word $w = w_0 w_1 \ldots \in \Sigma^\omega$ in the language of $\mathcal{A}$, there exists an accepting run $\pi = q_0 q_1 \ldots \in Q^\omega$ with corresponding color sequence $\rho = \rho_0 \rho_1 \ldots \in \mathbb{N}^\omega$ for it such that for all $j \in \mathbb{N}$, we have $(q_{j+1}, \rho_j) = f(w_0 \ldots w_j)$. Note that such a run is unique for each w.

Given an automaton $\mathcal{A} = (Q, \Sigma, \delta, Q_0)$, we say that a tuple $(\tilde{Q}, \tilde{\delta})$ with $\tilde{Q} \subseteq Q$ and $\tilde{\delta} \subseteq \delta \cap \tilde{Q} \times \Sigma \times \tilde{Q} \times \mathbb{N}$ is a *strongly connected component* (SCC) if, for each $q, q' \in \tilde{Q}$, we have that there exists a sequence of states $q_1, \ldots, q_n$ all in $\tilde{Q}$ for some $n \in \mathbb{N}$ such that $q_1 = q$, $q_n = q'$, and for every $1 \leq j < n$, there exist $x \in \Sigma$ and $c \in \mathbb{N}$ such that $(q_j, x, q_{j+1}, c) \in \tilde{\delta}$. We say that $(\tilde{Q}, \tilde{\delta})$ is a *maximal SCC* if no states and transitions can be added without losing the property that the (resulting) tuple is an SCC. Transitions that can only be taken once in a run (starting from any state) are called *transient*; they connect different SCCs. We also say that a state is *transient* if it can only be visited once along a run. For some co-Büchi automaton $\mathcal{A}$, we say that some SCC $(\tilde{Q}, \tilde{\delta})$ of $\mathcal{A}$ is an *accepting SCC* if $\tilde{\delta} \subseteq \tilde{Q} \times \Sigma \times \tilde{Q} \times \{2\}$. We furthermore call $(\tilde{Q}, \tilde{\delta})$ a *maximal accepting SCC* if $(\tilde{Q}, \tilde{\delta})$ cannot be strictly extended with further states or accepting transitions without losing the property that it is an accepting SCC.

3 Towards A Canonical Language Representation

The core definition we provide in this paper, namely the *natural color of a word*, lifts the idea of parity acceptance from deterministic automata to languages. Such a natural color will always be defined with respect to a given language, but for the brevity of presentation, we will not always mention this language henceforth.

Since, at the level of languages, there are no colors of transitions that can be referred to, the definition of the natural color of a word needs to capture the idea of colors in a way that does not employ the colors of transitions.

In this section, we make some observations on why some languages need a certain number of colors in a deterministic parity automaton, and identify ways in which we can abstract from the automaton representation along the way. We then distil the observations to define the natural color of a word in the next section.

Niwiński and Walukiewicz [10] have given a polynomial-time algorithm to minimize the number of different colors in a deterministic parity automaton. While their algorithm targets parity automata in which states – rather than transitions – are labeled with colors, it is not difficult to extend it to transition-based acceptance.

The core idea used in their algorithm is that, in order for a deterministic parity automaton that recognizes a language to need at least n colors, there needs to exist a so-called *flower* with at least n colors. Such a flower is defined to satisfy two properties. It firstly has a sequence of colors $c_1 < c_2 < \ldots < c_n$ such that every two successive colors in the sequence alternate by whether they are even or odd. Secondly, there exists a center state such that, for each color c_i, there exist paths from the center state back to itself such that the dominating color occurring along the transitions along the path is c_i. Following the terminology of Niwiński and Walukiewicz, we refer to such paths as *flower loops*. Figure 1 shows an example parity automaton that contains such a flower over the colors 1, 2, 3, 4, and 5.

Niwiński and Walukiewicz have shown that no deterministic parity automaton with fewer than n colors can encode a language that admits a flower with n colors [10]. This is because a flower defines a hierarchy over words that are, alternatingly, accepted or rejected by an automaton, and the n different colors are needed to detect on which level in the hierarchy a word is located.

Figure 2 shows such a hierarchy of words for the parity automaton from Figure 1. They all have in common that the state q_c in the center of the flower is visited infinitely often in a run of the automaton for the respective word.

The first word, $(ca)^\omega$, leads to only transitions with color 5 being taken, so the color of the word is 5, and the word is rejected.

The second word is built by injecting bb strings at positions of the word at which the respective run is in the center state q_c. Note that bb causes transitions $q_c \xrightarrow{b} q_2 \xrightarrow{b} q_c$ in the run for the second word, so that the added string causes an excursion in the run that leads back to the same state. In this way, the run of the second word can be obtained from the run for the first word by adding elements at the positions in which a finite substring is inserted into the word. Because in the run for the second word, color 4 is visited infinitely often, this becomes the color of the modified word.

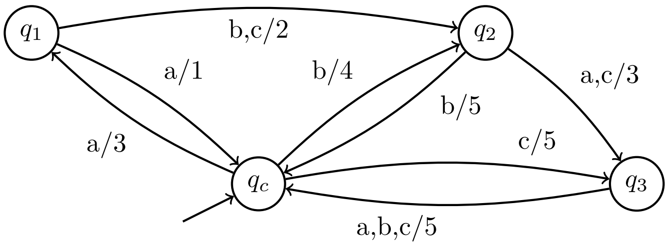

Figure 1 A flower in a parity automaton. The flower loops are $q_c \xrightarrow{a} q_1 \xrightarrow{a} q_c$ for color 1, $q_c \xrightarrow{a} q_1 \xrightarrow{c} q_2 \xrightarrow{b} q_c$ for color 2, $q_c \xrightarrow{b} q_2 \xrightarrow{a} q_3 \xrightarrow{c} q_c$ for color 3, $q_c \xrightarrow{b} q_2 \xrightarrow{b} q_c$ for color 4, and $q_c \xrightarrow{c} q_3 \xrightarrow{c} q_c$ for color 5.

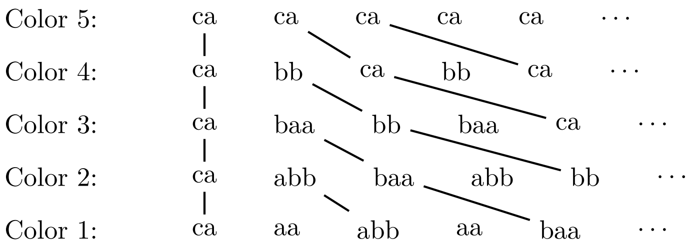

Figure 2 An example hierarchy of words for the parity automaton from Figure 1, used in an example in Section 3. The words from the lower colors are obtained from those of higher colors by inserting language invariant words, which are flower loops in the given example. The lines show where the flower loops from the words with higher color are in the words with lower color.

The other words are built according to the same idea: By injecting finite words leading from q_c back to q_c (while taking a transition with a lower color) into the word at positions in the word on which a run for the word is at state q_c anyway, we obtain a new word that is accepted by the automaton if, and only if, the old word is rejected.

The most significant color (here: 1) now has the special property that inserting more loops does not change whether or not the word is accepted, as the most significant color of the automaton is already visited along the run. We can formalize this as follows: Let $w = w_0 w_1 \ldots$ be a word and $\pi = q_0 q_1 \ldots$ be a run of the automaton over w. We say that a finite word w' is a state-invariant injection at a position $i \in \mathbb{N}$ in the word if $\delta(q_i, w') = q_i$. We can characterize the words w that are recognized with the most significant color in a strongly connected component (SCC) of the parity automaton as those for which every word $\tilde{w}$ that results from an infinite sequence of injections of state-invariant words into w is accepted by the automaton if, and only if, w is accepted by the automaton. Note that the restriction to strongly connected components is necessary, as other parts of a deterministic parity automaton may employ more colors, and hence this property holding for a word whose run ends in some SCC does not exclude that some other automaton part uses more colors, including a lower one.

3.1 The Case of the Most Significant Color

Let us now distill the definition of acceptance with the most significant color to the language case. In the resulting definition, the restriction to a single SCC will also be lifted.

The hierarchy of words from Figure 2 refers to the states of a given deterministic parity automaton: Finite words can only be inserted at places in which the added finite words loop from the state in which the run of the automaton is at the insertion place, back to the same state. This idea can be lifted by replacing the notion of state by *suffix language invariance*. We say that inserting a finite word $u \in \Sigma^*$ at position $i \in \mathbb{N}$ in a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ is a suffix language invariant injection into w for a language $L \subseteq \Sigma^\omega$ if $\mathcal{L}^{\mathit{suffix}}(L, w \ldots w_i) = \mathcal{L}^{\mathit{suffix}}(L, w \ldots w_i u)$.

In this definition, the suffix languages take the role of the flower center states, and they have the nice property that they are independent of an automaton representation of the language.

Injecting any finite number of loops from a state back to itself into a run of a parity automaton does not change the color with which the respective word is accepted. It makes sense to expect for the definition of the natural color of a word that similarly, any finite number of suffix language invariant injections should not change the color of a word.

With this in mind, we can try to liberate the definition regarding which words should be recognized with the most significant color from any reference to a particular automaton: they are those words for which an infinite number of suffix language invariant word injections does not change whether or not a word is accepted. It is, however, necessary to carefully define where exactly in a word these injections can be made.

To see this, consider the case of the language "there are infinitely often two as in a row". It can be recognized by a deterministic *Büchi* automaton, i.e., a deterministic parity automaton with colors 0 and 1. Since 0 is the most significant color, all words in the language need to have this natural color. This language has only a single suffix language, namely itself. If we would require for a word to be of natural color 0 that *all* infinite sequences of suffix language invariant word injections result in an accepted word, then *no* word would be accepted with this natural color: This is because this would include injecting a b as every second letter in a word. Consequently, not all injection sequences need to be tolerated for a word to be of natural color 0. But it also does not suffice if only *some* injection is tolerated: In an extreme case, that includes injecting the empty word everywhere, which never affects acceptance.

A compromise between these two extremes is to use different quantifiers for the points of injection and the words being injected. We declare those words to have a lowest natural color, for which *there exists* an infinite sequence of points, at which suffix invariant words can be injected, such that, *for all* insertions of sequences of suffix invariant words at these points, the resulting word is accepted if, and only if, the original word was. While this solves the problem from the short example in the paragraph above, it is not trivially clear whether or not other problems remain when using this definition. The correctness of the construction from the next section, however, shows that this is precisely the definition we need.

3.2 Generalizing to All Colors

To generalize the idea of the natural color of a word from the most significant color to the general case, we can follow an inductive argument and – in a sense – peel the language off, layer by layer. We look at the colors $c \in \mathbb{N}$ in ascending order and define, for each color $c+1$, which words are natural for this color, under the assumption that we have such a definition for colors up to c. To do so, we can marry an inductive definition of what constitutes the color of a word in a deterministic parity automaton with the automaton-agnostic definition of the natural color of a word for the most significant color from above.

We start with revisiting an inductive definition of the color of a word in a deterministic parity automaton. We can characterize the words accepted with color 0 to be those along whose runs transitions with color 0 are taken infinitely often. For colors $c > 0$, we can define that a word is accepted with color c if transitions with color c are taken infinitely often for its run, *and* the word is not accepted with a color smaller than c. The nice property of this rather indirect definition is that it only refers to colors already defined and a single additional color.

In an orthogonal composition of this idea for an inductive definition with the central idea from the previous subsection, we can allocate the color c as the natural color of a word w if there exists an infinite set of indices such that, for every sequence of suffix invariant strings inserted at these indices, we have that the resulting word $\tilde{w}$

- *either* has a natural color smaller than c,
- *or* it does not, and $\tilde{w}$ is in the language if, and only if, w is in the language.

Thus, we only require that inserting the words makes no difference regarding acceptance where the resulting words are not of a smaller natural color.

This definition has the nice property that, by induction, for every color, the natural colors of words are uniquely defined purely by the language of the word, without reference to an automaton representation. The concrete definition given in the next section, however, makes the words for colors $c+i$ (for $i \geq 0$) contained in the words for a color c, so we define the natural color to be the *minimal* color to which this definition is applicable.

While it is, again, not trivially clear that this idea captures all necessary aspects of the natural color of a word, our results in the following section show that this is the case.

4 The Natural Color of a Word

In this section, we define our notion of the natural color of a word (with respect to a given language) based on the observations from the previous section. Based on this definition, we show how a sequence of co-Büchi automata can be obtained from the deterministic parity automaton such that, after the minimization of the co-Büchi automata, the sequence is a canonical representation of the language. This construction has two steps, namely first *streamlining* the deterministic parity automaton, and then extracting the co-Büchi automata from it. Using this sequence of automata as representation for the natural colors of words (with respect to the represented language), we finally show that the natural colors of words can be read off from the streamlined deterministic parity automaton directly, which shows that the natural colors of words can be read off from *any* deterministic parity automaton after preprocessing it.

We start with defining the colors in which a word is "at home" for a given language. The natural color of a word is then the minimal color in which a word is at home. First, we repeat some concepts from the previous section to make Definition 1 below self-contained. We say that some finite word $u \in \Sigma^*$ is *suffix language invariant* for a language L after a finite word $w \in \Sigma^*$ if we have $\mathcal{L}^{\mathit{suffix}}(L, w) = \mathcal{L}^{\mathit{suffix}}(L, wu)$.

Let a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ over an alphabet Σ be given. We say that some word $w' \in \Sigma^\omega$ is the result of a suffix language invariant injection of a sequence of words $u_0, u_1, \ldots$ at positions $J = \{i_0, i_1, \ldots\}$ with $i_0 < i_1 < \ldots$ in w if, and only if, $w' = w_0 w_1 \ldots w_{i_0}\, u_0\, w_{i_0+1} \ldots w_{i_1}\, u_1\, w_{i_1+1} \ldots w_{i_2}\, u_2 \ldots$ and for each $j \in \mathbb{N}$, we have that u_j is suffix language invariant for $w_0 \ldots w_{i_j}$.

▶ **Definition 1.** *For every even/odd i, we say that a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ is* at home in color *$i \in \mathbb{N}$ for some language L if there exists an infinite subset $J \subseteq \mathbb{N}$ such that, for every possible sequence of finite words $u_0, u_1, \ldots$, if a word w' is the result of a suffix language invariant injection of $u_0, u_1, \ldots$ at positions J, then we have that*

- *w' is already at home in a color strictly smaller than i, or*
- *both w and w' are in L and i is even, or both w and w' are not in L and i is odd.*

Note that the first case in the preceding definition cannot apply for color $i = 0$ (as there is no smaller color), so only the second case is of relevance for $i = 0$.

We call the minimal natural number that a word w is at home in the *natural color of* w (for a given language L).

As a small remark, this definition could also be given in a variant where the lowest color a word can be at home in is 1, which swaps the order in which we check for "i is even" and "i is odd". This affects the number of different natural colors that the words can have (for a given language) by at most 1. All results given henceforth also work with odd and even colors swapped.

The color of a run of a DPA is not necessarily the natural color of the word defined above (for the language of the DPA). We will, however, show how a deterministic parity automaton can be used to define a family of good-for-games co-Büchi automata that can determine the natural color of a word.

4.1 Streamlining DPAs

The first concept to use is the concept of a *structured parity automaton*: We call a DPA *structured* if (1) all states of $\mathcal{A}$ are reachable and (2) if two states q and q' are equivalent, then they are in the same maximal SCC.

Turning a given DPA into an equivalent structured DPA is cheap.[1] First, non-reachable states are removed. Then, an arbitrary minimal preorder that preserves reachability among the maximal SCCs of the DPA is defined. Two states are equivalent according to this preorder if, and only if, they are reachable from each other (i.e., they are in the same maximal SCC). Apart from this, the preorder follows the reachability relation in that, if a state q is reachable from a state q', then $q \geq q'$.

A transition to a state q with a language equivalent state q' such that $q' > q$ is then re-directed to some language equivalent state q'' that is maximal according to this preorder.

As the position in this preorder can only grow along every run, there are only finitely many re-directions taken on every run. The language of the automaton is not changed by this operation.

Finally, the states that become unreachable by rerouting the transitions are removed again to make the resulting automaton structured.

▶ **Definition 2.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ be a structured DPA. We define its* streamlined *version to be the outcome of the following* streamlining process*, in which the structure of $\mathcal{A}$ is not changed, but a new color is assigned to each transition.*

We first produce a copy of the structure of $\mathcal{A}$, creating a fresh coloring graph *$\mathcal{G} = (Q, \Sigma, \delta)$ (which is called "coloring graph" because it is used for determining the new colors in $\mathcal{A}$; $\mathcal{G}$ itself does not have colors). We will then successively remove states and transitions from $\mathcal{G}$ (not from $\mathcal{A}$), while assigning the transitions new colors in $\mathcal{A}$.*

Starting with $i = 0$, we do the following until $\mathcal{G}$ is empty.

1. *We first partition $\mathcal{G}$ into maximal SCCs.*
2. *We then identify all transient transitions in $\mathcal{G}$. We change their colors in $\mathcal{A}$ to i, and then remove the transient states and transitions from $\mathcal{G}$.*
3. *We check if in any maximal SCC of $\mathcal{G}$, the least color of the transitions (in $\mathcal{A}$) between states in the SCC has the same parity as i.*
   - *If such transitions are found, we first change their colors in $\mathcal{A}$ to i, remove them from $\mathcal{G}$, and then return to (1) without incrementing i.*
   - *If there are no such transitions, we increment i and go back to (1).*

These steps are repeated until $\mathcal{G}$ is empty.

---

[1] The procedure can, for instance, be found in [11]; while it is only described for Büchi and co-Büchi automata there, it can also be applied to parity automata, as done in [9].

The purpose of the construction is to iteratively *lower* the colors of transitions in $\mathcal{A}$ towards the most significant one whenever that is possible without changing the language of the automaton. The *structure* of the automaton is not altered. The algorithm identifies transitions that can be recolored to a lower color i because they can only be taken finitely often along runs that do not have a dominating color lower than or equal to i anyway. Also, the algorithm identifies transitions that can be safely recolored to i because, while changing the color to i changes the dominating color of some runs, it never changes the parity of the dominating color.

The streamlining construction above is a variant of an algorithm by Carton and Maceiras [4] to relabel the colors of states in a deterministic parity automaton with state-based acceptance. The construction has been adapted to the case of transition-based acceptance and gives rise to a more concise correctness argument stated next. It retains the structured automaton's property that, for each set of language equivalent states, all states in the set can be found in the same maximal SCC.

▶ **Lemma 3.** *The streamlining process terminates and does not change the language of $\mathcal{A}$.*

Proof. We first observe that, by a simple inductive argument, before i is incremented, all transitions with color $\leq i$ have been removed from $\mathcal{G}$.

As induction basis, for $i = 0$, every transition with color 0 that remains after step (2) is the minimal color in their maximal SCC, and therefore removed in step (3).

For the induction step, after incrementing the counter to $i+1$, all transitions with color $\leq i$ have been removed by the induction hypothesis, such that all remaining transitions with color $i+1$ must either be transient (and removed in step (2)), or of minimal color in a maximal SCC (and are then removed in step 3).

The algorithm always terminates because the color of a transition is only ever changed once (as the transition is removed from $\mathcal{G}$ whenever their color in $\mathcal{A}$ is changed). When no color is (re)assigned in $\mathcal{A}$ and removed from $\mathcal{G}$ in an iteration, i is increased. Finally, once i exceeds the number of colors of $\mathcal{A}$, the graph $\mathcal{G}$ must be empty, leading to termination.

The induction argument above also establishes that the color of each transition can only be *reduced* by this construction, but to no color lower than any (new) color of the edges that have previously been removed from $\mathcal{G}$.

This observation is the basis for establishing language equivalence. We establish this language equivalence step-wise, considering only the effect that the changes of colors initiated by step (2) or step (3) in one iteration have on the dominating color of *some* (arbitrary) run of $\mathcal{A}$.

For colors changed in step (2) of the algorithm, we observe that, if the respective transition t occurs infinitely often, some other transition that has previously been removed from $\mathcal{G}$ must occur infinitely often, too. The color of any of these removed transitions is $\leq i$, such that changing the color of t to i does not change the dominating color of the run. If t however occurs only finitely often along the run in $\mathcal{A}$, it cannot change the dominating color of the run either.

For a transition t whose color is changed in step (3), we distinguish three cases. First, if t occurs only finitely often along the run under consideration, the color change does not influence the dominating color of the run. Second, if the run eventually remains in the same maximal SCC (in $\mathcal{G}$) as t was and t occurs infinitely often, then the previous color of t was the dominating color of the run, because t's color was minimal among the colors of the SCC. But then the new dominating color is i, which has the same parity as (and is no greater than) the previous color of t. Finally, if the run does not eventually get stuck in the maximal SCC of t in $\mathcal{G}$, but t occurs infinitely often, then there are infinitely many transitions passed that have been re-colored before t, and that therefore have a color $\leq i$, such that the re-coloring of t does not change the dominating color of the run. ◀

4.2 From Streamlined DPAs to Color-Recognising GCAs

We will not relate the colors of the runs in streamlined automata directly to the natural color of a word, but use them to define good-for-games co-Büchi automata (*GCAs*) that do so. These GCAs are easy to obtain from $\mathcal{A}$.

▶ **Definition 4.** *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ be a streamlined automaton and i be a color that occurs in $\mathcal{A}$, then $\mathcal{A}_i = (Q, \Sigma, \delta_i, q_0)$ is the automaton such that, for all $(q, x, q', c) \in \delta$,*

- $(q, x, q', 2) \in \delta_i$ *if* $c \geq i$,
- $(q, x, q', 1) \in \delta_i$ *if* $c < i$, *and*
- $(q, x, q'', 1) \in \delta_i$ *for all* $q'' \in Q$ *with* $q' \sim_{\mathcal{A}} q''$ *such that, for all colors* c', $(q, x, q'', c') \notin \delta$.

The co-Büchi automata are defined such that, for all colors i, $\mathcal{A}_i$ accepts those words for which the run in $\mathcal{A}$ has a dominating color of at least i (using transitions of the first two types in the list above only). However, $\mathcal{A}_i$ accepts some additional words: The transitions added by the third item in the list above allow a run to "jump" to any state that is language-equivalent in $\mathcal{A}$ (if the transition is not already part of $\mathcal{A}_i$ by the first two items). In accepting runs, this can only happen finitely often, though.

We will show in the remainder of this subsection that, for all i, $\mathcal{A}_i$ is a good-for-games co-Büchi automaton that accepts exactly the words that have a natural color of at least i (with respect to the language of $\mathcal{A}$).

The first observation that we will use to achieve this goal is that the languages of these co-Büchi automata are obviously shrinking with growing index, simply because the transitions are the same, but some of the accepting transitions become rejecting transitions (i.e., their color changes from 2 to 1).

▶ **Observation 5.** *For $i \leq j$, $\mathcal{L}(\mathcal{A}_i) \supseteq \mathcal{L}(\mathcal{A}_j)$ holds. Also, $\mathcal{A}_0$ accepts all words as it only has accepting transitions (since no color is smaller than 0) while including outgoing transitions for each state/letter pair (as $\mathcal{A}$ is deterministic).*

▶ **Theorem 6.** *For all i, $\mathcal{A}_i$ is good-for-games.*

The proof is a pretty standard proof for co-Büchi automata: it essentially says "follow the run that has longest been through accepting transitions".

Proof. If the automaton $\mathcal{A}_i$ has no accepting run for a word $w = w_0 w_1 w_2 \ldots$, there is nothing to show.

We now assume that w has an accepting run $\pi = q_0 q_1 q_2 \ldots$, where j is the first position in π such that, for all $k \geq j$, $(q_k, w_k, q_{k+1}, 2)$ are accepting transitions.

We argue that $\mathcal{A}_i$ can accept w with the strategy to

1. follow accepting transitions where possible; note that there is at most one outgoing accepting transition for every state/letter pair, so this selection is deterministic, and
2. if no such transition is available when reading w_k, move to a state q'_{k+1} such that there exists a run prefix $\pi' = q_0 q'_1 q'_2 \ldots q'_{k+1}$ for $w_0 \ldots w_k$ with some $l < k+1$ such that the transitions taken from q'_l onwards are all accepting. In particular, state q'_{k+1} is chosen for some *lowest possible* value of l among such run prefixes (the way to choose ex aequo does not matter).

To see why this strategy yields an accepting run, consider a total order over all possible finite run prefixes. The prefixes are ordered by their size (starting from the smallest one), but otherwise the total order is arbitrary.

Whenever the strategy can continue along an accepting transition, it does so. When it has to take a rejecting transition after having read a finite prefix w' of w, it chooses a smallest run prefix ρ' such that w' has a unique finite run $\rho'\rho''$, where ρ'' contains only accepting transitions. The run $\rho'\rho''$ ends in some state q, and our strategy is to move to this state q.

The prefix $q_0 \ldots q_j$ is somewhere in this order, say at position p. Now, if there are at least p rejecting transitions taken when following the strategy, then $q_0 \ldots q_j$ will eventually be tried. From this point onward, no rejecting transitions are taken anymore in the run for w. If fewer than p rejecting transitions are taken when following the strategy, the resulting run is accepting as well.

Thus, we always have a strategy that only relies on the past, and $\mathcal{A}_i$ is good-for-games. ◀

▶ **Theorem 7.** *$\mathcal{A}_i$ accepts a word w if, and only if, the natural color of w for the language $\mathcal{L}(\mathcal{A})$ is at least i.*

Proof.

Induction basis: For $i = 0$, every word is accepted by $\mathcal{A}_0$.

Induction step: Let us assume that the property holds for all $i' < i$ for some $i > 0$.

For the induction step, we split the “if and only if” in the claim into its two directions. We first show that (*substep 1*) a word $w = w_0w_1 \ldots$ with natural color at most $i - 1$ is rejected by $\mathcal{A}_i$, and then argue that (*substep 2*) a word rejected by $\mathcal{A}_i$ has a natural color of at most $i - 1$. Taking both directions together and considering the remaining words (those *accepted* by $\mathcal{A}_i$ rather than those *rejected* my $\mathcal{A}_i$), we obtain that a word is accepted by $\mathcal{A}_i$ if, and only if, its natural color is at least i, which is to be proven.

Substep 1: Here, we show that a word $w = w_0w_1 \ldots$ with natural color of at most $i - 1$ is rejected by $\mathcal{A}_i$. If the natural color of w is strictly smaller than $i - 1$, then this follows directly from the inductive hypothesis and Observation 5. For the case of the natural color of w being exactly $i - 1$, we assume for contradiction that w has a natural color of $i - 1$ and w is accepted by $\mathcal{A}_i$. Using the definitions for acceptance by a co-Büchi automaton and the natural color of a word, we have that

- $\pi = q_0q_1q_2 \ldots$ is an accepting run of $\mathcal{A}_i$ on w,
- $p \in \mathbb{N}$ is a position such that, for all $j \geq p$, $(q_j, w_j, q_{j+1}, 2) \in \delta_i$ is an accepting transition in $\mathcal{A}_i$, and
- $J \subseteq \mathbb{N}$ is an infinite index set such that injecting suffix language invariant words at the positions in J always results in a word w' that
  (c1) has natural color that is strictly smaller than $i - 1$ *or*
  (c2) is accepted by $\mathcal{A}$ if, and only if, $i - 1$ is even,
  where we assume w.l.o.g. $j \geq p$ for all $j \in J$.

The accepting run π has, from position p onward, only transitions that have color of at least i in $\mathcal{A}$. Let $p' \geq p$ be the position from which these transitions are all in the same maximal accepting SCC S in $\mathcal{A}_i$. By the assumption that $\mathcal{A}$ is streamlined (which is a precondition to applying Definition 4), the maximal accepting SCC (of $\mathcal{A}_i$) has a transition that has a corresponding transition of color i in $\mathcal{A}$. To see this, note that we can only have an accepting SCC in $\mathcal{A}_i$ if it is also an SCC in the graph $\mathcal{G}$ built by the streamlining construction when starting to consider color i (as all transitions from and to states not in $\mathcal{G}$ at that point of the construction have been assigned colors strictly smaller than i in the streamlined automaton).

But then, either the minimal transition color in the SCC has the same parity as i, and then it is lowered to i in the streamlining construction, or it does not. In the latter case, the streamlining construction lowers the color of such a minimal transition color in the SCC to $i-1$ by the third step of the construction before actually considering color i, contradicting the assumption that the SCC has only accepting transitions in $\mathcal{A}_i$ (as transitions with color smaller than i are not accepting in $\mathcal{A}_i$ by Def. 4).

Since we now know that the minimal color in the maximal accepting SCC in $\mathcal{A}_i$ is i in the streamlined automaton $\mathcal{A}$, we can insert into w, in every position in $j \in J$, a suffix language invariant string, whose partial run is a cycle in both $\mathcal{A}_i$ and $\mathcal{A}$, in the latter case with minimal color i. Therefore, the resulting word w' is accepted by $\mathcal{A}$ if, and only if, i is even. As i and $i-1$ have a different parity, condition (c2) cannot hold for w'.

However, condition (c1) also cannot hold: The accepting run of $\mathcal{A}_i$ on w' is also an accepting run of $\mathcal{A}_{i-1}$, so its natural color is at least $i-1$ by our inductive hypothesis. Taking the falsification of both (c1) and (c2) together, we obtain that w cannot have a natural color of at most $i-1$. (contradiction)

Substep 2: Finally, we show that a word $w = w_0 w_1 \dots$ that is rejected by $\mathcal{A}_i$ has natural color of at most $i-1$. If the word is rejected by $\mathcal{A}_{i-1}$, then the natural color is at most $i-2$ by our inductive hypothesis, and there is nothing more to be shown. So we henceforth only need to consider the case that w is accepted by $\mathcal{A}_{i-1}$ but not $\mathcal{A}_i$. We define a suitable infinite set of indices $J \subseteq \mathbb{N}$.

We first assume for contradiction that there is a position $p > 0$ in the word such that, for all positions $p' > p$, there is a run $\pi = q_0 q_1 \dots$ of $\mathcal{A}_i$ where $(q_j, w_j, q_{j+1}, 2) \in \delta_i$ is, for all $p \leq j < p'$, an accepting transition in $\mathcal{A}_i$. If no such position exists, the finitely branching tree of runs of $\mathcal{A}_i$ on w that is pruned at all non-accepting positions after level p is infinite, and therefore has an infinite path. (contradiction to $w \notin \mathcal{L}(\mathcal{A}_i)$)

Using this observation, we fix an infinite ascending chain $0 < p_0 < p_1 < p_2 \dots$, such that, for all $j \geq 0$, no run has only accepting transitions in any segment $q_{p_j} q_{p_j+1} \dots q_{p_{j+1}}$.

We note that inserting suffix language invariant strings in positions of $J = \{p_j \mid j \in \mathbb{N}\}$ does not change that these segments have this property; consequently, any word w' that results from such insertions is still rejected by $\mathcal{A}_i$. We fix such a word w'.

Let c be the maximal color such that $w' = w'_0 w'_1 w'_2 \dots$ is in the language of $\mathcal{A}_c$. As $w' \notin \mathcal{L}(\mathcal{A}_i)$, $c < i$. If $c < i-1$, then its natural color is $c < i-1$ by induction hypothesis. If it is $c = i-1$, then the natural color must be at least $i-1$ by induction hypothesis. Moreover, $\mathcal{A}_{i-1}$ has an accepting run $\pi' = q_0 q'_1 q'_2 \dots$ on w'. Let p be a position in this run such that, for all $j > p$, the transitions $(q'_j, w'_j, q'_{j+1}, 2) \in \delta_{i-1}$ are accepting transitions of $\mathcal{A}_{i-1}$. Noting that π' is a rejecting run of $\mathcal{A}_i$, this entails that the lowest color that occurs infinitely often in the run $q'_p q'_{p+1} q'_{p+2} \dots$ of $\mathcal{A}_{q'_p}$ on $w'_p w'_{p+1} w'_{p+2} \dots$ is $i-1$. Thus $w'_p w'_{p+1} w'_{p+2} \dots$ is accepted by $\mathcal{A}_{q'_p}$, and therefore w' is accepted by $\mathcal{A}$ if, and only if, i is odd.

This concludes the proof that the natural color of w is $i-1$. ◀

Taking the results above together, we have obtained a construction for a language recognised by a given deterministic parity automaton that provides a sequence of co-Büchi automata that encode which word is at home in which color. We first compute a structured form of this deterministic parity automaton, then streamline it (Def. 2), and finally split the resulting parity automaton into the co-Büchi automata according to Def. 4. All three steps can be implemented to run in time polynomial in the size of the input automaton. Furthermore, since the state spaces of the co-Büchi automata are the same as the one in the parity automaton, they cannot be larger than the original parity automaton.

Since the split into co-Büchi automata is canonical, and the co-Büchi automata themselves can be made canonical (and minimal) using the existing polynomial-time construction from Abu Radi and Kupferman [1], we overall obtain a canonical representation of the language that the deterministic parity automaton we started with represents. Moreover, in can be computed in polynomial time.

4.3 Reading off Natural Colors from Streamlined Automata

The construction so far has the property that it does not immediately provide a direct way of computing the natural color of a word (yet). Given a word, we can check which of the co-Büchi automata built according to Def. 4 accepts the word to compute the natural color of a word (for the given language), but since they are good-for-games rather than deterministic, this is somewhat cumbersome.

The results above, however, allow us to also define a more direct way to determining the natural color of a word (with respect to a given language), as we show below as a side-result.

We define a *co-run* of a deterministic automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ on a word $w = w_0w_1w_2\ldots$ with a run $\pi = q_0q_1q_2\ldots$ as a sequence $\pi' = q_0q_1\ldots q_{p-1}q_pq'_{p+1}q'_{p+2}q'_{p+3}\cdots$ for some $p > 0$, such that $\pi'' = q'_pq'_{p+1}q'_{p+2}q'_{p+2}\ldots$ is the run of $\mathcal{A}_{q'_p}$ on the word $w' = w_pw_{p+1}w_{p+2}w_{p+2}\ldots$ for some state q'_p that is language equivalent to q_p ($q'_p \sim_{\mathcal{A}} q_p$).

The color of the set of co-runs for a word w is defined to be the maximal dominating color c that occurs infinitely often on some co-run of w.

▶ **Lemma 8.** *Let $\mathcal{A}$ be a streamlined automaton. Then the color of the set of co-runs of a word w is c if w is in the language of $\mathcal{A}_c$, but not in the language of $\mathcal{A}_{c+1}$.*

Proof. A co-run of $\mathcal{A}$ on w with dominating color c is an accepting run of $\mathcal{A}_c$.

An accepting run $\pi' = q'_0q'_1\ldots q'_{p-1}q'_pq'_{p+1}q'_{p+2}q'_{p+2}\ldots$ of $\mathcal{A}_{c+1}$ has some position p from which point onward only accepting transitions (which all have color $> c$ in $\mathcal{A}$) are taken. $\mathcal{A}$ therefore has a co-run $\pi' = q_0q_1\ldots q_{p-1}q_pq'_{p+1}q'_{p+2}q'_{p+2}\ldots$, whose dominating color is $> c$. (contradiction) ◀

By combining this lemma with the previous theorem, we get the following corollary.

▶ **Corollary 9.** *Let $\mathcal{A}$ be a streamlined automaton. Then the color of the set of co-runs of a word w is its natural color for $\mathcal{L}(\mathcal{A})$.*

Co-runs are closely related to the GCAs we have defined earlier. The difference is that the "new" transitions to language equivalent states can be used only once along a run. This allows for having a definition on the deterministic automaton (without falling back on good-for-games automata), and is therefore simpler. It also binds the proofs together: the *minimal prefix* from the proof of Theorem 6 corresponds to the shortest prefix at which this single transition to a language equivalent state can be taken. While this provides a more direct connection to the color, the restriction to taking these transitions at most once loses the good-for-games property: as a wrong decision cannot be corrected, access to the remainder of the run may be required. This makes GCAs more attractive, as co-Büchi languages have canonical representatives.

Note that, for an ultimately periodic word (i.e., a word of the form $w = uv^\omega$ for $u, v \in \Sigma^*$), the highest color among the co-runs can be computed in the time polynomial in the number of states, which allows reading off the natural color of a word from a (streamlined) DPA without building the canonical representation of the language of the DPA.

5 Related Work

There already exists an indirect normal form of ω-languages. Every ω-regular language can be represented as a deterministic finite automaton (DFA): This DFA accepts words of the form $u\$v$ for which the *ultimately periodic word* uv^ω is in the ω-language to be represented. Calbrix et al. [3] showed how to compute such a DFA from a given nondeterministic Büchi automaton (to which a deterministic parity automaton is easy to translate). The minimized DFA for this *lasso language* over finite words can then serve as a canonical representation of the ω-language, as two automata representing ω-regular languages encode the same language if, and only if, they accept the same ultimately periodic words.

DFAs that capture ultimately periodic words have furthermore been refined to *families* of DFAs [2] that can be more succinct and share the property to consist of multiple sub-automata with the sequences of co-Büchi automata that we define in this paper.

Such DFAs (or families of DFAs), however, do not implement a core idea of automata, namely to read a word letter-by-letter and to encode the relevant information about the letters of the word already read in a state. It is also unknown how the sizes of such automata relate to the size of a minimal deterministic parity automaton representing the language. Finally, neither DFAs for lasso languages nor families of DFAs have a direct connection to the complexity of a language, as it is, for example, captured by the minimal number of colors used in deterministic parity automata.

6 Conclusion

A classical question in the theory of automata is how to define *canonical automata*.

In this paper, we have taken a step back, and looked at the question of how to define canonical automata for ω-regular languages from a new angle. For a language to be defined canonically by, say, a canonical deterministic parity automaton, a word first and foremost needs a natural color. What should this color be?

Picking the flowers of Niwiński and Walukiewicz [10] as a starting point, we have lifted the same principle to languages in a way that is oblivious to the automaton used.

For an ω-regular language L, we look at a sequence of languages $L_0 \supset L_1 \supset L_2 \ldots \supset L_c$ such that

- L_0 is the universal language;
- L_1 is the smallest co-Büchi language contained in L_0 and containing $\overline{L} \cap L_0$ such that $\overline{L_1}$ is closed under insertion;
- L_2 is the smallest co-Büchi language contained in L_1 and containing $L \cap L_1$ such that $\overline{L_2}$ is closed under insertion;
- L_3 is the smallest co-Büchi language contained in L_2 and containing $\overline{L} \cap L_2$ such that $\overline{L_3}$ is closed under insertion; etc.

The *closure under insertion* of a set $\overline{L_i}$ refers to the existence of an infinite set of positions (for each word) at which arbitrary suffix language invariant finite letter sequences can be added to the word without the resulting word leaving $\overline{L_i}$.

This assigns a *color* to each infinite *word* w, both in the language and outside of it, purely defined by the maximal i such that $w \in L_i$ – we say that w has a natural color of i (for L). As one would expect for a parity condition, $w \in L$ if, and only if, the natural color i of w is even.

The natural color of w for L is thus defined without reference to an automaton (or any other representation of the language). Yet, L is recognized by a deterministic parity automaton with maximal color c if, and only if, L_{c+1} is empty. This sets the minimal maximal color in the automaton to the maximal i such that L_i is non-empty, which further connects this construction to deterministic automata.

We infer these languages by turning a single *streamlined* deterministic parity automaton, which is cheap and easy to obtain from any deterministic parity automaton that recognizes L: L_i contains the set of words whose co-runs have colors of at least i.

Beyond providing evidence *that* a word is in the language, it also provides insight into *why* it is part of this language by peeling off co-Büchi languages of accepted and rejected words layer by layer.

Returning to our chain of languages, this answers the "why co-Büchi?" question that begs to be asked. Each L_c is a co-Büchi language, which is an ideal basis for a natural representation, because co-Büchi languages have recently obtained a canonical representation, albeit not for deterministic automata, but for *good-for-games co-Büchi automata* with transition-based acceptance (GCAs). We can use this to obtain a natural representation for the languages that allow us to identify the natural color of a word.

References

1 Bader Abu Radi and Orna Kupferman. Minimizing GFG transition-based automata. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 100:1–100:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.100.

2 Dana Angluin and Dana Fisman. Learning regular omega languages. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory - 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8776 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2014. doi:10.1007/978-3-319-11662-4_10.

3 Hugues Calbrix, Maurice Nivat, and Andreas Podelski. Ultimately periodic words of rational w-languages. In *9th International Conference on Mathematical Foundations of Programming Semantics (MFPS)*, volume 802 of *Lecture Notes in Computer Science*, pages 554–566. Springer, 1993. doi:10.1007/3-540-58027-1_27.

4 Olivier Carton and Ramón Maceiras. Computing the Rabin index of a parity automaton. *RAIRO Theor. Informatics Appl.*, 33(6):495–506, 1999. doi:10.1051/ita:1999129.

5 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 395–410. Springer, 2006. doi:10.1007/11874683_26.

6 John Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971. doi:10.1016/B978-0-12-417750-5.50022-1.

7 Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2015. doi:10.1007/978-3-662-47666-6_24.

8 Christof Löding. Efficient minimization of deterministic weak omega-automata. *Inf. Process. Lett.*, 79(3):105–109, 2001. doi:10.1016/S0020-0190(00)00183-6.

9 Christof Löding and Andreas Tollkötter. State space reduction for parity automata. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, volume 152 of *LIPIcs*, pages 27:1–27:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CSL.2020.27.

10 Damian Niwiński and Igor Walukiewicz. Relating hierarchies of word and tree automata. In *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science, Paris, France, February 25-27, 1998, Proceedings*, volume 1373 of *Lecture Notes in Computer Science*, pages 320–331. Springer, 1998. doi:10.1007/BFb0028571.

11 Sven Schewe. Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.400.

12 Sven Schewe. Minimising good-for-games automata is NP-complete. In Nitin Saxena and Sunil Simon, editors, *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference)*, volume 182 of *LIPIcs*, pages 56:1–56:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FSTTCS.2020.56.