The Formal Language Theory Column

by

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20133 Milano, Italy
pighizzini@di.unimi.it

The contribution by Antonio Casares presented in this issue of BEATCS studies the placement of the acceptance condition on automata over infinite words and discusses how the community is currently experiencing a shift from using state-based acceptance to using transition-based acceptance, explaining why this shift is taking place. Maybe in the near future, for the reasons discussed in this survey, transition-based omega-automata will become the default model in the community. The survey is derived from the final chapter of Antonio's PhD Thesis.

Transition-based vs stated-based acceptance for automata over infinite words

Antonio Casares*

Abstract

Automata over infinite objects are a well-established model with applications in logic and formal verification. Traditionally, acceptance in such automata is defined based on the set of states visited infinitely often during a run. However, there is a growing trend towards defining acceptance based on transitions rather than states.

In this survey, we analyse the reasons for this shift and advocate using transition-based acceptance in the context of automata over infinite words. We present a collection of problems where the choice of formalism has a major impact and discuss the causes of these differences.

Contents

*University of Warsaw, Poland. Email: antoniocasaressantos@gmail.com
This work was supported by the Polish National Science Centre (NCN) grant "Polynomial finite state computation" (2022/46/A/ST6/00072).

1 Introduction

Automata theory is a central and long-established topic in computer science. The definition of finite automata has barely suffered any modification since the introduction of non-deterministic automata by Rabin and Scott [RS59]. However, the generalisation of automata to infinite words presents less stable definitions, as different modes of acceptance are best suited to different situations. Recently, there has been a shift in the community towards using transitions instead of states to encode the acceptance condition of ω-automata. In this survey, we analyse the reasons for this shift and advocate using transition-based acceptance in the context of automata over infinite words.

Automata over infinite words. An *automaton* over an input alphabet Σ is given by

- a finite set of states Q,
- a set of transitions $\Delta \subseteq Q \times \Sigma \times Q$,
- a set of initial states $Q_{\mathsf{init}} \subseteq Q$, and
- an acceptance condition.

A *run* over a (finite or infinite) word w is a path in the automaton starting in Q_{init} and with transitions labelled by the letters of w. The acceptance condition is thus a representation of the set of paths that are accepting.

If the automaton works over finite words, the acceptance condition usually takes the form of a subset of final states: a run is accepting if it ends in one of them (see Section 5 for further discussions on finite words). For automata over infinite words the situation is more complicated. Several acceptance conditions are commonly used, but they differ in expressive power and the complexity of related problems (see for instance [Bok18]). The main focus of this paper is the following dichotomy: Should we use states or transitions to encode the acceptance condition of automata over infinite words? More formally, we will consider acceptance conditions of one of the following forms.

A *state-based acceptance condition* is a language $Acc \subseteq Q^\omega$. A *transition-based acceptance condition* is a language $Acc \subseteq \Delta^\omega$.Usually, we represent them via a finite set of colours C, a colouring function $\gamma\colon Q \to C$ (resp. $\gamma\colon \Delta \to C$) and a language $Acc' \subseteq C^\omega$. That is, we see automata as transducers $\Sigma^\omega \to C^\omega$, and the acceptance condition is given by a subset of the image. Two languages that are commonly used as acceptance conditions are:

- Buchi = $\{w \in \{-, \bullet\}^\omega \mid w \text{ contains } \bullet \text{ infinitely often}\}$. We may refer to states (resp. transitions) coloured with $\bullet$ as *accepting*.
- coBuchi = $\{w \in \{-, \times\}^\omega \mid w \text{ contains } \times \text{ finitely often}\}$.

We show examples of Büchi automata in Figure 1.

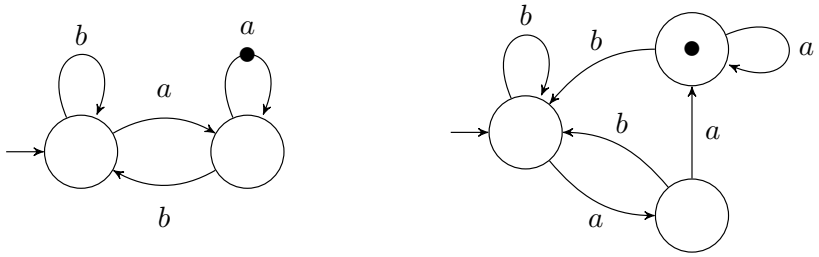

Figure 1: Two Büchi automata recognising the language of words containing infinitely many factors '*aa*'. The automaton on the left uses transition-based acceptance, while the automaton on the right is state-based.

The origins. Automata over infinite words were first introduced by Büchi in the 60s [Büc62], using a formalism that put the acceptance condition over states.[1] The tradition of employing state-based acceptance persisted in all subsequent classic foundational works on ω-automata: Muller's paper at the origin of the Muller condition [Mul63], Landweber's study of the complexity of ω-regular languages [Lan69], McNaughton's works on ω-regular expressions [McN66] and infinite games [McN93], Rabin's decidability result of S2S [Rab69], Wagner's paper introducing a hierarchy of complexity [Wag79], etc. Following this tradition, virtually all handbooks and surveys about automata on infinite objects use state-based acceptance [Eil74, Tho90, Tho97, GTW02, PP04, BK08, Kup18, BCJ18, WS21, Löd21, EB23]. To the best of our knowledge, the only exceptions are the recent book *Games on Graphs* edited by Fijalkow [Fij+25], and the book *An Automata Toolbox* by Bojańczyk [Boj].

The rise of transition-based acceptance. Automata with "effects" on transitions, such as sequential transducers[2] [Sha48, Sect.8][Mea55, Sch61a]

[1] Corroborating this claim can be quite challenging. The use of state-based acceptance can be observed, for instance, in the first line of the proof of Lemma 12 (page 8). In Büchi's 1969 paper with Landweber [BL69a], this is a bit simpler to appreciate in the definitions of Sup$Z$ and U, in the second page of the paper.

[2] Transducers with output on states were also considered by Moore [Moo56]. However, the model with output on transitions popularized by Mealy [Mea55] rapidly became the norm.

or weighted automata [Sch61b] have been considered since the beginnings of automata theory. Transition-based ω-automata made their first, though modest, appearance in the mid-80s. To the best of our knowledge, their first occurrences were in Michel's work on the connection between *Linear Temporal Logic* (LTL) and automata [Mic84], and in Kurshan's paper on the complementation of deterministic Büchi automata [Kur87].[3] In the early 90s, Le Saëc made more systematic use of this model [Saë90, SPW91, VSL95]. He reintroduced transition-based Muller automata under the name of table-transition automata, and characterised which languages admit a unique *morphism-minimal* Muller automaton: those that can be recognised by a Muller automaton with one state per residual of the language [VSL95, Cor. 5.15]. This characterisation no longer holds for state-based automata (see Example 10 for an illustration on how the previous property is sensitive to the placement of the acceptance condition). Despite the works of Le Saëc, transition-based automata were used only scarcely in the following years.

Some notable exceptions to the predominant use of state-based automata in the 2000s are given by a series of works concerning the translation of LTL formulas to automata. In 1999, Couvreur proposed a translation using transition-based generalised Büchi automata [Cou99]. A similar algorithm was the base for the tool `ltl2ba` by Gastin and Oddoux [GO01] (the importance of the use of transition-based automata in this work is discussed in [GL02]). The use of transition-based acceptance in this subarea was further fostered by the tool Spot [DP04, Dur+22], influenced by Couvreur's approach. More recently, transition-based automata have been adopted in the HOA format [Bab+15], and it is the primary model in other tools such as Owl [KMS18] or `ltl3tela` [Maj+19]. We refer to [Dur07, pages 66-67] for an overview of the use of state-based and transition-based approaches to the translation of LTL prior to 2007.

A turning point occurred in 2019, as Abu Radi and Kupferman proved that transition-based history-deterministic coBüchi automata can be minimised in polynomial time [AK19], while Schewe showed that the corresponding problem is NP-complete for state-based automata [Sch20]. Since then, there is an increasing interest for transition-based ω-automata, and, as discussed in Sections 3 and 4, many recent results rely on the use of this model.

Why was the use of state-based acceptance widespread? We may wonder why state-based automata were the ubiquitous model for more than 50 years. Probably the most influential factor is that ω-automata generalise automata over finite words, for which acceptance over states is a natural choice. Some construc-

[3] The possibility of using transition-based acceptance was previously suggested in [Par81, Section 8.2]. Some sources [Red99] mention that transition-based acceptance was already suggested by Redziejowski in 1972 [Red72]; unfortunately we could not get access to this paper.

tions of ω-automata build on automata over finite words, and for some of these, state-based acceptance appears naturally.

One such example is the characterisation of languages recognised by deterministic Büchi automata as limits of languages of finite words [Lan69]. A language $L \subseteq \Sigma^\omega$ can be recognised by a deterministic Büchi automaton if and only for some regular language of finite words $L_{\mathsf{fin}} \subseteq \Sigma^*$ we have:

$$L = \overrightarrow{L_{\mathsf{fin}}} = \{w \in \Sigma^\omega \mid w \text{ contains infinitely many prefixes in } L_{\mathsf{fin}}\}.$$

Building a state-based Büchi automaton from a deterministic automaton recognising L_{fin} is easy: we just need to interpret the final states of the automaton as accepting Büchi states. The converse direction follows similarly.

Structure of the survey. We start by showing in Section 2 that we can switch between state and transition-based acceptance with at most a linear blow-up. However, we already notice a key difference: going from a state-based automaton to a transition-based one does not require adding any additional state, while deciding the minimal number of states required to perform the converse transformation is NP-hard (Proposition 3). In Sections 3 and 4, we study problems on ω-automata and games where the choice between transition-based and state-based acceptance may strongly affect the complexity of a given problem. In Section 5 we explore transition-based acceptance for automata over finite words. Finally, in Section 6 we discuss some of the reasons causing the striking differences between the two models.

Definitions are introduced progressively as needed. The reader may use the hyperlinks on technical terms to quickly see their definition.

2 From states to transitions and vice versa

At first sight, it could seem that there is no great difference between state-based or transition-based acceptance: we can go from one model to the other with at most a linear blow-up. However, transition-based automata are always smaller, and going from a state-based automaton to a transition-based one in an optimal way is NP-hard, as stated in Proposition 3.

Proposition 1. *Every state-based automaton can be relabelled with an equivalent transition-based acceptance condition.*

Proof. Let $Acc \subseteq Q^\omega$ be the acceptance condition of the automaton, and let $\gamma\colon \Delta \to Q$ be the function assigning to each transition (q, a, q') its source state q. Then, (γ, Acc) is an equivalent transition-based acceptance condition. □

In general, we cannot relabel in a similar manner a transition-based automaton to obtain an equivalent state-based one. We can, however, build an equivalent state-based automaton paying a small blow-up on the number of states.

Proposition 2. *Every transition-based automaton admits an equivalent state-based automaton with at most* $|Q||\Delta| + |Q_{\mathsf{init}}|$ *states.*

Proof. Let $\mathcal{A}$ be a transition-based automaton with acceptance $Acc \subseteq \Delta^\omega$. We define the automaton having:

- States: $(Q \times \Delta) \cup Q_{\mathsf{init}}$.
- Transitions: For every transition $t' = q \xrightarrow{a} q'$ in $\mathcal{A}$, we let $(q,t) \xrightarrow{a} (q',t')$, and $q \xrightarrow{a} (q',t')$ if $q \in Q_{\mathsf{init}}$.
- Initial states: Q_{init}.
- Acceptance condition: We define $\gamma\colon Q \to \Delta \cup \{x\}$ by: $\gamma(q,t) = t$ and $\gamma(q_0) = x$ if $q_0 \in Q_{\mathsf{init}}$. The acceptance condition is given by the colouring γ and the language $x \cdot Acc$.

It is immediate to check that the obtained automaton is equivalent to $\mathcal{A}$. □

In both proofs above, the obtained automaton is not only equivalent to the original one, but there is a bijection between the runs of both. We formalise this idea with the notion of *locally bijective morphisms* [Cas+24, Def.3.3].

Given two automata $\mathcal{A}, \mathcal{A}'$ over the same alphabet, a *locally bijective morphism* is a function $\varphi\colon Q \to Q'$ such that:

- $\varphi(Q_{\mathsf{init}}) = Q'_{\mathsf{init}}$,
- for all $(q,a,q') \in \Delta$, $(\varphi(q),a,\varphi(q')) \in \Delta'$,
- for all $(p,a,p') \in \Delta'$ and $q \in \varphi^{-1}(p)$, there is $q' \in \varphi^{-1}(p')$ such that $(q,a,q') \in \Delta$, and
- a run ρ in $\mathcal{A}$ is accepting if and only if $\varphi(\rho)$ is accepting in $\mathcal{A}'$.

Intuitively, if $\varphi\colon \mathcal{A} \to \mathcal{A}'$ is a locally bijective morphism, it means that $\mathcal{A}$ has been obtained from $\mathcal{A}'$ by duplicating some of its states, for instance, via a product construction. For example, the automaton on the right of Figure 1 admits a locally bijective morphism to the automaton on its left.

Proposition 1 implies that for every state-based automaton there is a transition-based automaton of same size admiting a locally bijective morphism to it (the automaton itself). However, in the other direction, deciding whether there is a small state-based automaton admiting a locally bijective morphism towards a given transition-based automaton is hard, already for Büchi automata.

Proposition 3. *The following problem is* NP-complete*:*

Input: *A transition-based Büchi automaton $\mathcal{A}_{\text{tr}}$ and a positive integer n.*
Question: *Is there a state-based Büchi automaton with n states admitting a locally bijective morphism to $\mathcal{A}_{\text{tr}}$?*

Proof. To show NP-hardness, we use the reduction from Vertex Cover given by Schewe to show the NP-completeness of the minimisation of state-based deterministic Büchi automata [Sch10].

Let $G = (V, E)$ be an undirected graph. Consider the Büchi automaton $\mathcal{A}_G$ over the alphabet $\Sigma = V$ with states $Q_G = V$, all of them initial, and transitions $u \xrightarrow{v} v$ for every $(u, v) \in E$, and for $u = v$. For the Buchi condition, all transitions are accepting except the self-loops $v \xrightarrow{v} v$. This automaton recognises the paths in G, allowing repetition of vertices, but that visit at least two different vertices infinitely often.

Let k be the size of a minimal vertex cover of G. We claim that there is a state-based Büchi automaton with $|V|+k$ states admitting a locally bijective morphism to $\mathcal{A}_G$, and that this is optimal. To obtain such a state-based automaton, we duplicate every state v that is part of a given vertex cover. Let $v_\bullet, v_-$ be the two copies of this state, and set $v_\bullet$ to be an accepting state. Among non-duplicated states, transitions are as in $\mathcal{A}_G$. For duplicated states, we let $v_i \xrightarrow{v} v_-$ for $i \in \{-, \bullet\}$ and $u_i \xrightarrow{v} v_\bullet$ for $(u, v) \in E$. It is easy to chech that $\varphi(v_i) = v$ defines a locally bijective morphism.

For the converse direction, let $\mathcal{A}$ be a state-based Büchi automaton and $\varphi\colon \mathcal{A} \to \mathcal{A}_G$ a locally bijective morphism. For every state v in $\mathcal{A}_G$, $\varphi^{-1}(v)$ must contain a non-accepting state, as a run ending in v^ω is rejecting in $\mathcal{A}_G$. We claim that the set of vertices such that $\varphi^{-1}(v)$ contains an accepting state is a vertex cover of G. Indeed, for every edge $(u, v) \in E$, a word ending in $(uv)^\omega$ is accepting in $\mathcal{A}_G$, therefore, either $\varphi^{-1}(u)$ or $\varphi^{-1}(v)$ contains an accepting state.

The problem is in NP, as there is always such an automaton with $2|Q|$ states. For $n < 2|Q|$, it suffices to guess an automaton $\mathcal{A}_{\text{st}}$ with n states and a locally bijective morphism $\varphi\colon \mathcal{A}_{\text{st}} \to \mathcal{A}_{\text{tr}}$. □

In our opinion, the above propositions indicate that state-based acceptance is often innapropriate. We believe that, in an ideal scenario, each state of a minimal automaton should stand for some semantic properties of the language they represent (in the case of automata over finite words, these are the residuals of the language). This cannot be the case for state-based ω-automata, as some states must be allocated to encode parts of the acceptance condition.

3 Minimisation and transformations of automata

In this section we study three problems relating to ω-automata: minimisation, conversion of acceptance condition and determinisation. We discuss how the use of transition-based or state-based acceptance can critically affect these problems.

3.1 Minimisation of coBüchi automata

The *minimisation problem* asks, given an automaton and a number n, whether there is an equivalent automaton with at most n states. This problem admits different variants, depending on the class of automata that constitutes the search space (here we assume that this class is the same for the input and output automata).

In 2010, Schewe showed that the minimisation problem is NP-hard for most types of deterministic state-based ω-automata, including Büchi, coBüchi or parity [Sch10]. It came as a surprise when Abu Radi and Kupferman showed that history-deterministic coBüchi automata can be minimised in polynomial time [AK22] (conference version from 2019 [AK19]). Soon after, Schewe showed that the same problem is NP-hard for state-based automata.[4]

An automaton is *history-deterministic* (abbreviated HD) if there is a resolver $\sigma\colon \Sigma^*\times\Sigma \to \Delta$, such that for every word w accepted by the automaton, the run over w built following the transitions given by σ is accepting. History-deterministic coBüchi automata are as expressive as deterministic ones, but they can be exponentially more succinct [KS15].

Proposition 4 ([AK22],[Sch20])**.** *History-deterministic transition-based coBüchi automata can be minimised in polynomial time.*

The minimisation problem for history-deterministic state-based coBüchi automata is NP-complete.

The work of Abu Radi and Kupferman provided the basis of many subsequent results, including new representations for ω-regular languages [ES22, Ehl25], minimisation of HD generalised coBüchi automata [Cas+25], passive learning of HD coBüchi automata [LW25] and characterisations of positional languages [CO24]. The transition-based assumption is essential to all these works.

Schewe's proof of NP-hardness of the minimisation of deterministic state-based Büchi automata [Sch10] strongly relies on putting the acceptance over states. In fact, as we have seen in Proposition 3, what this reduction shows is that finding a minimal state-based automaton that simulates a transition-based one is

[4] Note that the critical difference lies in the output class, as we can convert the input from state-based to transition-based in polynomial time.

NP-hard. It was not until 2025 that the minimisation of deterministic transition-based Büchi and coBüchi automata was shown to be NP-hard, requiring a highly technical proof [AE25].

3.2 Translation from Muller to parity

The complexity of the acceptance condition used by an automaton may greatly affect the computational cost of dealing with these automata. Namely, many problems are PSPACE-hard for Muller automata [HD05], but become tractable for parity automata [Cal+22, Bok18]. Therefore, an important task is to simplify the acceptance condition of a given automaton. In practice, this usually takes the following form: given an automaton using a Muller condition, build an equivalent automaton using a parity condition.

The parity and Muller conditions are defined as follows:

- $\mathsf{parity}(d) = \{w \in \{1, \dots, d\}^\omega \mid \liminf w \text{ is even}\}$.
- $\mathsf{Muller}(\mathcal{F}) = \{w \in C^\omega \mid \mathtt{Inf}(w) \in \mathcal{F}\}$, for $\mathcal{F} \subseteq \mathcal{P}(C)$ a family of subsets and $\mathtt{Inf}(w)$ the set of colours that appear infinitely often in w.

Recently, an optimal transformation has been introduced – based on a structure called the *Alternating Cycle Decomposition* (ACD) – transforming a Muller automaton $\mathcal{A}$ into a parity one [Cas+24]. Formally, it produces a transition-based parity automaton that admits a locally bijective morphism to $\mathcal{A}$ and with a minimal number of states among parity automata admiting such a morphism. This transformation can be performed in polynomial time provided that the ACD can be computed efficiently; this is the case for example if the acceptance condition of $\mathcal{A}$ is generalised Büchi, defined as follows:

- $\mathsf{genBuchi} = \{w \in \mathcal{P}(C)^\omega \mid \bigcup_{A \in \mathtt{Inf}(w)} A = C\}$.

Proposition 5 (Follows from [Cas+24, Thm. 5.35])**.** *Given a generalised Büchi automaton $\mathcal{A}$, we can build in polynomial time a transition-based Büchi automaton admiting a locally bijective morphism to $\mathcal{A}$ that has a minimal number of states among Büchi automata admitting locally bijective morphisms to $\mathcal{A}$.*

However, the optimality result of the ACD-transformation strongly relies on the use of transition-based acceptance in the output automaton, as the previous problem becomes NP-hard for state-based automata.

Proposition 6. *The following problem is* NP-complete*:*

Input: *A state-based generalised Büchi automaton $\mathcal{A}$ and an integer n.*

Question: *Is there a state-based Büchi automaton with n states admitting a locally bijective morphism to $\mathcal{A}$?*

Proof. We can use the same reduction as in the proof of Proposition 3 (which in turn comes from [Sch10]). Indeed, we can replace the transition-based Büchi condition of the automaton $\mathcal{A}_G$ by a state-based generalised Büchi condition. □

3.3 Determinisation of Büchi automata

The determinisation of Büchi automata is a fundamental problem in the theory of ω-automata, studied since the introduction of the model [Büc62]. The first asymptotically optimal determinisation construction is due to Safra [Saf88], which transforms a Büchi automaton into a deterministic Rabin one. In 1999, Redziejowski proposed a variant for building a transition-based automaton from a given ω-regular expression [Red99]. Later on, Piterman [Pit06] and Schewe [Sch09] further improved Safra's construction, reducing the number of states of the final automaton (see also [Red12]). Schewe's construction transforms a Büchi automaton of size n into a deterministic Rabin automaton of size at most $\mathsf{sizeDet}(n)$, which is naturally equipped with a transition-based acceptance condition (with $\mathsf{sizeDet}(n) = o((1.65n)^n)$). In 2009, Colcombet and Zdanowski [CZ09] showed that the Piterman-Schewe construction is tight (up to 0 states!) as we precise now.

Proposition 7 ([CZ09])**.** *There exists a family of Büchi automata $\mathcal{A}_n$ with n states, such that a minimal transition-based deterministic Rabin automaton equivalent to $\mathcal{A}_n$ has $\mathsf{sizeDet}(n)$ states.*

We can obtain a state-based automaton by augmenting the number of states, but doing so we no longer have a matching lower bound. No such tight bounds are known for the determinisation of Büchi automata towards state-based automata.

The complementation and determinisation problems for Büchi and generalised Büchi automata with transition-based acceptance were further studied by Varghese in his PhD Thesis [Var14]. In the works of Schewe and Varghese [SV12, SV14], they point out the suitability of transition-based acceptance for the study of transformations of automata.

4 Games on graphs and strategy complexity

A *game* is given by a directed graph $G = (V, E)$ with a partition of vertices into those controlled by a player Eve and those controlled by a player Adam, a initial vertex and a *winning condition* defined in the same way as the acceptance condition of automata (which can be state-based or transition-based). The players move

a token in turns producing an infinite path, and Eve wins if this path belongs to the winning condition.

An important concept with applications for the decidability of logics [BL69b, GH82] and verification [BCJ18] is that of strategy complexity: how complex is it to represent a winning strategy? The simplest kind of strategies are *positional* ones. A strategy is *positional* if it can be represented by a function $\sigma\colon V \to E$: when in a vertex v controlled by Eve, she plays the transition $\sigma(v)$. More generally, a strategy is said to use *finite-memory* if the choice at a given moment only depends on a finite amount of information from the past, or, said differently, it can be implemented by a finite automaton (we refer to [Fij+25, Section 1.5] for formal definitions).

As already noticed by Zielonka [Zie98], and as we will see next, strategy complexity is quite sensitive to the placement of the winning condition.

4.1 Bipositionality over infinite games

We say that a language $Win \subseteq C^\omega$ is *positional* if for every game with winning condition Win, if Eve has a winning strategy, she has a positional one. A language Win is *bipositional* if both Win and its complement are positional, or, said differently, if both Eve and Adam can play optimally using positional strategies. Depending on whether we consider games with transition-based or state-based winning condition, we will say accordingly *positional over transition/state-based games*.

A celebrated result in the area is the proof of bipositionality of parity languages [EJ91, Mos84]. In 2006, Colcombet and Niwiński proved that these are the only prefix-independent bipositional languages over infinite game graphs [CN06], establishing an elegant characterisation of bipositionality. As indicated in the title of their paper, this characterisation only holds for transition-based games.

Proposition 8 ([CN06])**.** *A prefix-independent language $Win \subseteq C^\omega$ is bipositional over transition-based games if and only if there is $d \in \mathbb{N}$ and a mapping $\phi\colon C \to \{1, \dots, d\}$ such that $w \in Win$ if and only if $\phi(w) \in \mathsf{parity}(d)$.*

Proposition 9 ([Zie98, Section 6])**.** *There is a prefix-independent language that is bipositional over totally-coloured state-based games, but is not equivalent to $\mathsf{parity}(d)$ for any d.*

Proof sketch. An example of such a language is

$$Win = \{w \in \{a, b\}^\omega \mid \text{ both } a \text{ and } b \text{ appear infinitely often in } w\}.$$

Intuitively, if Eve is in a vertex coloured a, she can follow a strategy leading to a vertex coloured b in a positional way (and vice-versa).

From Adam's point of view, if he can win, there are some vertices from which he can force to never produce 'a' or force to never produce 'b' (and this can be done positionally). Removing those vertices, we define a positional strategy recursively. (Note that this can also be done for transition-based games, in fact, from Adam's point of view, *Win* is a Rabin condition, which are positional.) □

The characterisation of bipositionality was generalised to all (not necessarily prefix-independent) languages in [CO24, Thm. 7.1]. A necessary condition for bipositionality is that the language should be recognised by a transition-based deterministic parity automaton with one state per residual of the language. This property is very sensitive to the placement of the acceptance condition, if suffices to consider the language Buchi that cannot be recognised by a state-based automaton with a single state. The next example shows another version of this.

Example 10. *Consider the language*

$$L = \{w \in \{a, b\}^\omega \mid \textit{if letter 'a' occurs in } w \textit{ then it appears infinitely often}\}.$$

This language has two residuals: $\varepsilon^{-1}L$ *and* $a^{-1}L$*. It can be recognised by a transition-based parity automaton (even a Büchi automaton) with two states, as shown in Figure 2. One can check that it also satisfies the other conditions from [CO24, Thm. 7.1], so it is bipositional. However, it is not possible to recognise L with a state-based parity automaton with only* 2 *states.*

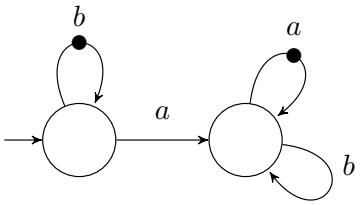

Figure 2: A Büchi automaton recognising the bipositional language of words that either contain no a, or infinitely many a's. This automaton has one state per residual of the language. A state-based parity automaton recognising this language must have at least 3 states.

4.2 Positionality via monotone graphs

Recently, Ohlmann characterised positionality by means of *monotone universal graphs* [Ohl23]. Not only this characterisation concerns positionality over

transition-based games, but the main notion of monotone graph radically uses the colouring on transitions. An ordered edge-coloured graph is *monotone* if whenever $v \xrightarrow{a} u$, $v \leq v'$ and $u' \leq u$, then the edge $v' \xrightarrow{a} u'$ also appears in the graph. Such kind of properties can only be naturally phrased in edge-coloured graphs.

Universal monotone graphs have been used to study the algorithmic complexity of solving different types of games on graphs, such as parity and mean-payoff [Col+22], and the above characterisation has been generalised to the memory of languages [CO25a].

4.3 The memory of ω-regular languages

The *memory* of a language *Win* is the minimal $m \in \mathbb{N}$ such that in any game with objective *Win*, if Eve has a winning strategy, she has one implemented by an automaton with at most m states. A result with major implications in logic is the fact that ω-regular languages have finite-memory [BL69b, GH82].

Recently, Casares and Ohlmann gave an effective way of computing the memory of ω-regular languages [CO25b], based on a characterisation using the notion of *ε-completable parity automata*. The definition of this notion is rooted in the use of transition-based acceptance: A parity automaton is *ε-completable* if for every pair of states q, q' and even colour x of the parity condition, we can either add a transition $q \xrightarrow{\varepsilon:x} q'$ or a transition $q' \xrightarrow{\varepsilon:x+1} q$ without modifying the language recognised by the automaton.

In 2023, Bouyer, Randour and Vandenhove showed that ω-regular languages are exactly those that are arena-independent finite-memory determined (that is, both Eve and Adam admit finite automata implementing strategies in every game with winning condition *Win*) [BRV23, Thm. 7]. The use of transition-based acceptance is key for the construction of a parity automaton recognising a language with the above property [BRV23, Section 5].

In 2021, Casares showed that the smallest automata that can be used for implementing winning strategies in every game using a given Muller language $\mathsf{Muller}(\mathcal{F})$ are exactly deterministic Rabin automata recognising $\mathsf{Muller}(\mathcal{F})$ [Cas22, Thm. 27]. In a related work, Casares, Colcombet and Lehtinen showed that the memory of $\mathsf{Muller}(\mathcal{F})$ coincides with the number of states of a minimal history-deterministic Rabin automaton recognising this language [CCL22, Thm. 5]. Both results only apply to transition-based Rabin automata.

5 What about finite words?

In light of the results above, one naturally wonders whether a shift to transition-based acceptance would also be beneficial for automata on finite words (*DFAs* in the following). Classical finite automata have a robust mathematical theory – notably, every regular language admits a canonical minimal DFA – and state-based acceptance is the undisputed preferred option for them. However, transition-based variants have been considered recently in works about synthesis of LTL over finite traces [Shi+20, Xia+21, Xia+24, Dur+25] and about translations of regular expressions over valuations of atomic propositions [MRD24].

Definition of acceptance. One option to define the acceptance of transition-based finite automata is simply to specify a set of final transitions: a run is accepting if its last transition belongs to this set.[5] More generally, following the definition of ω-automata used in this document, we define the *acceptance condition of a transition-based DFA* as a language $Acc \subseteq \Delta^*$: a run is accepting if it belongs to Acc. If Acc is a regular language, such an automaton accepts a regular language (we can convert it into a classical DFA by a product construction). Using a colouring function $\gamma\colon \Delta \to \{-, \circledcirc\}$ as in the introduction, we can recast acceptance by final transitions as automata using the following condition:

$$Acc_{\text{Last}} = \{w \in \{-, \circledcirc\}^* \mid \text{the last letter of } w \text{ is } \circledcirc\}.$$

The role of prefix-independence and the empty word. When using the above general model of transition-based DFAs we encounter one inconvenience: the language recognised starting from a given state q may be ill-defined, since the set of runs accepted from q depends on the particular path that led to q from the initial state. Independence from the past of the run is a key property, notably for defining a minimal DFA, where each state corresponds to a left-quotient of the language.

This problem would not arise if the acceptance condition Acc was *prefix-independent*, that is, if for all sequences of transitions u_0 and u:

$$u_0 u \in Acc \iff u \in Acc.$$

However, the only prefix-independent languages of finite words are the empty and the full language, which cannot be used to recognise non-trivial languages. Indeed, if Acc is prefix-independent, then $u \in Acc \iff \varepsilon \in Acc$ for all $u \in \Sigma^*$.

Nevertheless, the language Acc_{Last} is almost prefix-independent, as it satisfies:

$$\text{for all } u_0 \text{ and } u \neq \varepsilon, \quad u_0 u \in Acc_{\text{Last}} \iff u \in Acc_{\text{Last}}\,.$$

[5] In this case, we should also specify whether the empty word is accepted.

This property makes Acc_{Last} well-suited for state-based acceptance, as the acceptance of ε can be encoded in a state, obtaining a definition of acceptance that is agnostic to the way we reach a given state.

Minimal transition-based DFA. It is well-known that the minimal state-based DFA of a regular language $L \subseteq \Sigma^*$ is given by the equivalence classes of the Myhill-Nerode congruence. In order to fit the transition-based setting, we can coarsen this relation, disregarding separations by the empty word:

$$u \mathrel{\dot\sim_L} v \quad \overset{\text{def}}{\iff} \quad \text{for all } w \neq \varepsilon,\ uw \in L \iff vw \in L.$$

The next lemma is an easy check.

Lemma 11. *The relation $\mathrel{\dot\sim_L}$ is an equivalence relation over Σ^*. Moreover, if $u \mathrel{\dot\sim_L} v$, then $ua \mathrel{\dot\sim_L} va$ for all $a \in \Sigma$.*

In the following, by a transition-based DFA we mean one with acceptance by final transitions, that is, using the acceptance condition Acc_{Last}.

Proposition 12. *Every regular language of finite words has a unique minimal transition-based DFA, which has one state per equivalence class of $\mathrel{\dot\sim_L}$.*

Proof. Let $L \subseteq \Sigma^*$ be a regular language. Consider the DFA $\mathcal{A}_{\min}$ having as states the $\mathrel{\dot\sim_L}$-classes of L, with $[\varepsilon]$ the initial state, and transitions $[u] \xrightarrow{a} [ua]$, where accepting transitions are those with $ua \in L$. Moreover, we need to specify whether $\varepsilon \in L$; in the positive case, we let the initial transition of the automaton be accepting. This automaton is well-defined and recognises L thanks to Lemma 11.

Let $\mathcal{A}$ be a transition-based DFA recognising L. For a state q in $\mathcal{A}$, let

$$L_\varepsilon^{\mathcal{A}}(q) = \{w \in \Sigma^+ \mid \text{ the run over } w \text{ from } q \text{ is accepting}\}.$$

It holds that, if u labels a path from the initial state to q, then $L_\varepsilon^{\mathcal{A}}(q) = L_\varepsilon^{\mathcal{A}_{\min}}([u])$. Moreover, $L_\varepsilon^{\mathcal{A}_{\min}}(u) \neq L_\varepsilon^{\mathcal{A}_{\min}}([v])$ if $u \not\mathrel{\dot\sim_L} v$. Therefore, $\mathcal{A}_{\min}$ has at most as many states as $\mathcal{A}$, and in case of equality, they are isomorphic. □

Proposition 12 implies that transition-based DFAs are not larger than state-based ones. Moreover, they can be strictly smaller, as shown by the following example and Corollay 14 (see also [MRD24, Figs. 2-4]).

Example 13. *Let $\Sigma = \{a, b\}$ and consider the language of words that either have even lenght and end by 'b', or have odd length and end by 'a'. A minimal state-based DFA for this language, with 4 states, is given on the left of Figure 3. Note that the states q_a and q_b are not equivalent, as only one of them is accepting. However, $L_\varepsilon(q_a) = L_\varepsilon(q_b)$ (idem for p_a and p_b). Therefore, we can merge these states, obtaining a transition-based DFA with only 2 states.*

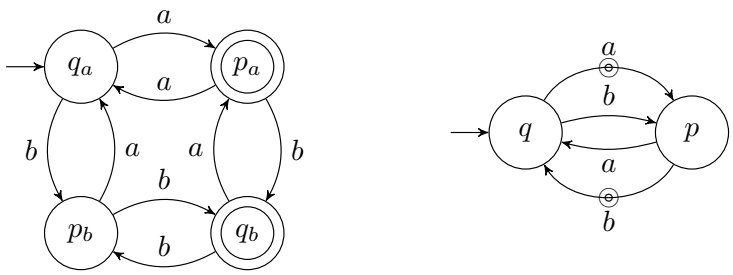

Figure 3: Two automata recognising the language $L = \{w \in \{a,b\}^+ \mid w \text{ is of even lenght if and only if it ends by '}b\text{'}\}$. The automaton on the left is the minimal state-based DFA of L, and the automaton on the right is its minimal transition-based DFA.

Generalising the previous example and using Proposition 12, we obtain:

Corollary 14. *Every transition-based DFA admits an equivalent state-based DFA with at most twice as many states. This bound is tight: there is a family of languages for which a minimal state-based DFA has twice as many states as a minimal transition-based DFA.*

Proof. For the first claim, it suffices to note that the index of the classical Myhill-Nerode congruence is at most twice the index of $\dot{\sim}_L$.

For the second claim, let $\Sigma = \{a,b\}$ and consider the language:

$$L_n = \{w \in \Sigma^+ \mid w \text{ ends by '}b\text{' if and only if } |w| \equiv 0 \bmod n\}.$$

The congruence $\dot{\sim}_L$ has n classes, corresponding to the remainder of $|w|$ modulo n. The Myhill-Nerode congruence has $2n$ classes, as words ending with a different letter are not equivalent. □

We note that such a gap does not appear for non-deterministic automata. Indeed, every transition-based NFA can be converted into an equivalent state-based NFA with only one more state. It suffices to add a sink state, which will be the only accepting state, and duplicate all accepting transitions redirecting one copy towards this sink. (This operation increases the number of transitions though.)

Where does this leave us? As we have seen, transition-based DFAs can be smaller than classical state-based ones. Moreover, most standard constructions adapt to the transition-based setting without problem (determinisation, product construction, removal of ε-transitions, etc). Some of them, such as the conversion of regular expressions, may even benefit from the use of transition-based acceptance [MRD24]. Transition-based DFAs can be of particular interest when used

as an intermediate step for the construction of ω-automata [MRD24], or when the acceptance of the empty word is irrelevant, as in LTL_f semantics.

However, there are signs pointing towards the canonicity of state-based acceptance for DFAs. Notably, the syntactic monoid of a language equals the transition monoid of its minimal state-based DFA [Pin, Prop. 4.28]. It is unclear to us what would be the correct way to recover the syntactic monoid from a transition-based DFA. Other questions regarding the transition-based model remain open. For instance, are there acceptance conditions other than Acc_{Last} that lead to unique minimal DFAs for all regular languages?

6 Outlook: Why all these differences?

We have seen various situations where transition-based acceptance is more advantageous, both for practical and theoretical reasons. The following question arises naturally: What are the fundamental differences between state-based and transition-based models that lead to such contrasting properties?

Composition of transitions. A basic operation at the heart of many reasonings in automata theory is *composition of transitions*. If an automaton contains transitions $p \xrightarrow{a} q$ and $q \xrightarrow{b} r$, one can go from p to r by reading ab, and any "effect" of this path should be the result of concatenating the effects of these two transitions. That is, a suitable automata model should allow to add the transition $p \xrightarrow{ab} r$. For automata over infinite words, the acceptance of the automaton obtained by adding this transition can only be defined in a sensible way by using a transition-based condition.

This composition operation is key for the celebrated connection between automata and algebra. The suitability of transition-based models for algebraic approaches is explicitly mentionned in Michel's work introducing transition-based ω-automata [Mic84, Section II]:

> *Using unstable graphs, instead of a set of nodes that must be traversed infinitely often, is better suited to the algebraic operations we will define [...]* [6]

Similarly, one of LeSaëc's motivations for the use of transition-based automata was to obtain an algebraic proof of McNaughton's theorem for infinite words [SPW91]. The Muller automaton obtained from a given semigroup is naturally transition-based, see [SPW91, page 18] and [Col11, Section 6].

[6] In French in the original: *L'utilisation de graphes instables au lieu d'un ensemble de nœuds dans lequel on doit passer infiniment souvent se prête mieux aux opérations algébriques que nous définirons [...].*

As mentioned in Section 4, composition of transitions is also essential in the fruitful approach for solving and analysing infinite duration games based on universal graphs, which relies on the notions of monotonicity, ε-completion and the technique of saturation (for the latter, see [CF18, Section 4], [Col+22, Section 4.1] or [Ohl23, Section 3.3]).

We note, however, that in the case of finite words, this does not provide strong evidence in favour of transition-based acceptance. Indeed, state-based DFAs also allow for composition of transitions, as the acceptance of a run is only determined by its final destination.

Paths in graphs. As explained in the introduction, an acceptance condition is a representation of a subset of paths in an automaton. A path in a graph is commonly defined as a sequence of edges. In fact, a sequence of vertices does not completely determine a path, as different paths may share the same sequence of vertices. This is the main reason why transition-based automata are more succinct than state-based ones.

Final thoughts

The collection of results presented in this survey indicates that, despite the fact that the size of state-based and transition-based automata only differ by a linear factor, transition-based models are easier to manipulate and have a nicer theory. We therefore advocate adopting transition-based acceptance as the default model for ω-automata.

We expect that the use of transition-based acceptance will ease the finding of automata-based characterisation of classes of languages. This has already been the case, for example, in the characterisation of positional ω-regular languages based on parity automata with a particular structure [CO24, Thm. 3.1].

In the same spirit, it appears that the use of transition-based models will be required for obtaining canonical models of automata over infinite words or trees. Steps in this direction have already been made [ES22, Ehl25, LW25], building on the description of canonical history-deterministic coBüchi automata by Abu Radi and Kupferman [AK22].

Acknowledgements. I warmly thank Thomas Colcombet for many discussions on the benefits of transition-based acceptance, Alexandre Duret-Lutz for valuable comments on automata over finite words and for sharing many historical references, Géraud Sénizergues for pointing me to the works of Bertrand Le Saëc and Pierre Ohlmann for helpful feedback on a draft of this paper.

References

[AE25] Bader Abu Radi and Rüdiger Ehlers. "Characterizing the polynomial-time minimizable ω-automata". In: *Corr* abs/2504.20553 (2025). DOI: 10.48550/ARXIV.2504.20553.

[AK22] Bader Abu Radi and Orna Kupferman. "Minimization and canonization of GFG transition-based automata". In: *Log. Methods Comput. Sci.* 18.3 (2022). DOI: 10.46298/lmcs-18(3:16)2022.

[AK19] Bader Abu Radi and Orna Kupferman. "Minimizing GFG transition-based automata". In: *ICALP*. Vol. 132. LIPIcs. 2019, 100:1–100:16. DOI: 10.4230/LIPIcs.ICALP.2019.100.

[Bab+15] Tomás Babiak, Frantisek Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Kretínský, David Müller, David Parker, and Jan Strejcek. "The Hanoi Omega-Automata format". In: *CAV*. Vol. 9206. 2015, pp. 479–486. DOI: 10.1007/978-3-319-21690-4_31.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. URL: https://mitpress.mit.edu/9780262026499/principles-of-model-checking/.

[BCJ18] Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. "Graph games and reactive synthesis". In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 921–962. DOI: 10.1007/978-3-319-10575-8_27.

[Boj] Mikołaj Bojańczyk. *An automata toolbox*. Version of 7 February, 2025. URL: https://www.mimuw.edu.pl/~bojan/papers/toolbox.pdf.

[Bok18] Udi Boker. "Why these automata types?" In: *LPAR*. Vol. 57. EPiC Series in Computing. 2018, pp. 143–163. DOI: 10.29007/c3bj.

[BRV23] Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. "Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs". In: *Theoretics* 2 (2023). DOI: 10.46298/theoretics.23.1.

[Büc62] J. Richard Büchi. "On a decision method in restricted second order arithmetic". In: *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science* (1962), pp. 1–11.

[BL69a] J. Richard Büchi and Lawrence H. Landweber. "Definability in the monadic second-order theory of successor". In: *J. Symb. Log.* 34.2 (1969), pp. 166–170. DOI: 10.2307/2271090.

[BL69b] J. Richard Büchi and Lawrence H. Landweber. “Solving sequential conditions by finite-state strategies”. In: *Transactions of the American Mathematical Society* 138 (1969), pp. 295–311. URL: http://www.jstor.org/stable/1994916.

[Cal+22] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. “Deciding parity games in quasi-polynomial time”. In: *SIAM Journal on Computing* 51.2 (2022), pp. 152–188. DOI: 10.1137/17M1145288.

[Cas22] Antonio Casares. “On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions”. In: *CSL*. Vol. 216. 2022, 12:1–12:17. DOI: 10.4230/LIPIcs.CSL.2022.12.

[Cas+24] Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. “From Muller to parity and Rabin automata: Optimal transformations preserving (history) determinism”. In: *TheoretiCS* Volume 3 (Apr. 2024). DOI: 10.46298/theoretics.24.12.

[CCL22] Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. “On the size of good-for-games Rabin automata and its link with the memory in Muller games”. In: *ICALP*. Vol. 229. 2022, 117:1–117:20. DOI: 10.4230/LIPIcs.ICALP.2022.117.

[Cas+25] Antonio Casares, Olivier Idir, Denis Kuperberg, Corto Mascle, and Aditya Prakash. “On the minimisation of deterministic and history-deterministic generalised (co)Büchi automata”. In: *CSL*. Vol. 326. 2025, 22:1–22:18. DOI: 10.4230/LIPICS.CSL.2025.22.

[CO25a] Antonio Casares and Pierre Ohlmann. “Characterising memory in infinite games”. In: *Log. methods comput. sci.* 21.1 (2025). DOI: 10.46298/LMCS-21(1:28)2025.

[CO24] Antonio Casares and Pierre Ohlmann. “Positional ω-regular languages”. In: *LICS*. ACM, 2024, 21:1–21:14. DOI: 10.1145/3661814.3662087.

[CO25b] Antonio Casares and Pierre Ohlmann. “The memory of ω-regular and $\mathrm{BC}(\Sigma^0_2)$ objectives”. In: *ICALP*. Vol. 334. 2025, 149:1–149:18. DOI: 10.4230/LIPICS.ICALP.2025.149.

[Col11] Thomas Colcombet. “Green’s relations and their use in automata theory”. In: *LATA*. Vol. 6638. 2011, pp. 1–21. DOI: 10.1007/978-3-642-21254-3_1.

[CF18] Thomas Colcombet and Nathanaël Fijalkow. “Parity games and universal graphs”. In: *Corr* abs/1810.05106 (2018). URL: http://arxiv.org/abs/1810.05106.

[Col+22] Thomas Colcombet, Nathanaël Fijalkow, Pawel Gawrychowski, and Pierre Ohlmann. “The theory of universal graphs for infinite duration games”. In: *Log. Methods Comput. Sci.* 18.3 (2022). DOI: 10.46298/lmcs-18(3:29)2022.

[CN06] Thomas Colcombet and Damian Niwiński. “On the positional determinacy of edge-labeled games”. In: *Theor. Comput. Sci.* 352.1-3 (2006), pp. 190–196. DOI: 10.1016/j.tcs.2005.10.046.

[CZ09] Thomas Colcombet and Konrad Zdanowski. “A tight lower bound for determinization of transition labeled Büchi automata”. In: *ICALP*. 2009, pp. 151–162. DOI: 10.1007/978-3-642-02930-1_13.

[Cou99] Jean-Michel Couvreur. “On-the-fly verification of linear temporal logic”. In: *World congress on formal methods in the development of computing systems*. Vol. 1708. 1999, pp. 253–271. DOI: 10.1007/3-540-48119-2_16.

[Dur07] Alexandre Duret-Lutz. “Contributions à l’approche automate pour la vérification de propriétés de systèmes concurrents”. PhD thesis. Université Pierre et Marie Curie (Paris 6), 2007. URL: https://www.lrde.epita.fr/~adl/dl/adl/duret.07.phd.pdf.

[DP04] Alexandre Duret-Lutz and Denis Poitrenaud. “SPOT: an extensible model checking library using transition-based generalized Büchi automata”. In: *MASCOTS*. IEEE Computer Society, 2004, pp. 76–83. DOI: 10.1109/MASCOT.2004.1348184.

[Dur+22] Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. “From Spot 2.0 to Spot 2.10: What’s new?” In: *CAV*. Vol. 13372. 2022, pp. 174–187. DOI: 10.1007/978-3-031-13188-2_9.

[Dur+25] Alexandre Duret-Lutz, Shufang Zhu, Nir Piterman, Giuseppe De Giacomo, and Moshe Y. Vardi. “Engineering an LTLf synthesis tool”. In: *CIAA*. To appear. 2025. URL: http://arxiv.org/abs/2507.02491.

[Ehl25] Rüdiger Ehlers. “Rerailing automata”. In: *Corr* abs/2503.08438 (2025). DOI: 10.48550/ARXIV.2503.08438.

[ES22] Rüdiger Ehlers and Sven Schewe. “Natural colors of infinite words”. In: *FSTTCS*. Vol. 250. 2022, 36:1–36:17. DOI: 10.4230/LIPIcs.FSTTCS.2022.36.

[Eil74] Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: https://www.worldcat.org/oclc/310535248.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. “Tree automata, mu-calculus and determinacy (extended abstract)”. In: *FOCS*. 1991, pp. 368–377. DOI: 10.1109/SFCS.1991.185392.

[EB23] Javier Esparza and Michael Blondin. *Automata theory: An algorithmic approach*. MIT Press, 2023. URL: https://michaelblondin.com/automata/.

[Fij+25] Nathanaël Fijalkow, C. Aiswarya, Guy Avni, Nathalie Bertrand, Patricia Bouyer, Romain Brenguier, Arnaud Carayol, Antonio Casares, John Fearnley, Paul Gastin, Hugo Gimbert, Thomas A. Henzinger, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Pierre Ohlmann, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, Mateusz Skomra, Nathalie Sznajder, and Pierre Vandenhove. *Games on graphs: from logic and automata to algorithms*. Ed. by Nathanaël Fijalkow. Cambridge University Press, 2025. URL: https://arxiv.org/abs/2305.10546.

[GO01] Paul Gastin and Denis Oddoux. “Fast LTL to Büchi automata translation”. In: *CAV*. Vol. 2102. 2001, pp. 53–65. DOI: 10.1007/3-540-44585-4_6.

[GL02] Dimitra Giannakopoulou and Flavio Lerda. “From states to transitions: improving translation of LTL formulae to Büchi automata”. In: *FORTE*. Vol. 2529. 2002, pp. 308–326. DOI: 10.1007/3-540-36135-9_20.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, eds. *Automata logics, and infinite games*. Springer, Berlin, Heidelberg, 2002. DOI: 10.1007/3-540-36387-4.

[GH82] Yuri Gurevich and Leo Harrington. “Trees, automata, and games”. In: *STOC*. 1982, pp. 60–65. DOI: 10.1145/800070.802177.

[HD05] Paul Hunter and Anuj Dawar. “Complexity bounds for regular games”. In: *MFCS*. 2005, pp. 495–506. DOI: 10.1007/11549345_43.

[KMS18] Jan Kretínský, Tobias Meggendorfer, and Salomon Sickert. “Owl: A library for ω-words, automata, and LTL”. In: *ATVA*. Vol. 11138. 2018, pp. 543–550. DOI: 10.1007/978-3-030-01090-4_34.

[KS15] Denis Kuperberg and Michał Skrzypczak. “On determinisation of good-for-games automata”. In: *ICALP*. 2015, pp. 299–310. DOI: 10.1007/978-3-662-47666-6_24.

[Kup18] Orna Kupferman. “Automata theory and model checking”. In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 107–151. DOI: 10.1007/978-3-319-10575-8_4.

[Kur87] R.P. Kurshan. “Complementing deterministic Büchi automata in polynomial time”. In: *Journal of computer and system sciences* 35.1 (1987), pp. 59–71. DOI: https://doi.org/10.1016/0022-0000(87)90036-5.

[Lan69] Lawrence H. Landweber. “Decision problems for omega-automata”. In: *Math. Syst. Theory* 3.4 (1969), pp. 376–384. DOI: 10.1007/BF01691063.

[Löd21] Christof Löding. “Automata on infinite trees”. In: *Handbook of automata theory*. Ed. by Jean-Éric Pin. 2021, pp. 265–302. DOI: 10.4171/AUTOMATA-1/8.

[LW25] Christof Löding and Igor Walukiewicz. “Minimal history-deterministic co-Buchi automata: Congruences and passive learning”. In: *Corr* abs/2505.14304 (2025). DOI: 10.48550/ARXIV.2505.14304.

[Maj+19] Juraj Major, Frantisek Blahoudek, Jan Strejcek, Miriama Sasaráková, and Tatiana Zboncáková. “Ltl3tela: LTL to small deterministic or nondeterministic Emerson-Lei automata”. In: *ATVA*. Vol. 11781. 2019, pp. 357–365. DOI: 10.1007/978-3-030-31784-3_21.

[MRD24] Antoine Martin, Etienne Renault, and Alexandre Duret-Lutz. “Translation of semi-extended regular expressions using derivatives”. In: *CIAA*. Vol. 15015. 2024, pp. 234–248. DOI: 10.1007/978-3-031-71112-1_17.

[McN93] Robert McNaughton. “Infinite games played on finite graphs”. In: *Annals of Pure and Applied Logic* (1993). DOI: 10.1016/0168-0072(93)90036-D.

[McN66] Robert McNaughton. “Testing and generating infinite sequences by a finite automaton”. In: *Information and control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X.

[Mea55] George H. Mealy. “A method for synthesizing sequential circuits”. In: *Bell Syst. Tech. J.* 34(5) (1955), pp. 1045–1079.

[Mic84] Max Michel. “Algebre de machines et logique temporelle”. In: *STACS*. 1984, pp. 287–298. DOI: 10.1007/3-540-12920-0_26.

[Moo56] Edward F. Moore. “Gedanken-experiments on sequential machines”. In: *Automata studies*. Ed. by C. E. Shannon and J. McCarthy. 34. 1956, pp. 129–153.

[Mos84] Andrzej W. Mostowski. “Regular expressions for infinite trees and a standard form of automata”. In: *SCT*. 1984, pp. 157–168. DOI: 10.1007/3-540-16066-3_15.

[Mul63] David E. Muller. “Infinite sequences and finite machines”. In: *Symposium on Switching Circuit Theory and Logical Design*. 1963, pp. 3–16. DOI: 10.1109/SWCT.1963.8.

[Ohl23] Pierre Ohlmann. “Characterizing positionality in games of infinite duration over infinite graphs”. In: *TheoretiCS* 2 (2023). DOI: 10.46298/theoretics.23.3.

[Par81] David Park. “Concurrency and automata on infinite sequences”. In: *Theoretical computer science*. Ed. by Peter Deussen. 1981, pp. 167–183. DOI: 10.1007/BFb0017309.

[PP04] Dominique Perrin and Jean-Eric Pin. *Infinite words - Automata, semigroups, logic and games*. Vol. 141. Pure and applied mathematics series. 2004.

[Pin] Jean-Eric Pin. *Mathematical foundations of automata theory. Notes of the MPRI lectures.* Version of March 2025. URL: https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf.

[Pit06] Nir Piterman. “From nondeterministic Büchi and Streett automata to deterministic parity automata”. In: *LICS*. 2006, pp. 255–264. DOI: 10.1109/LICS.2006.28.

[RS59] M. O. Rabin and D. Scott. “Finite automata and their decision problems”. In: *IBM journal of research and development* 3.2 (1959), pp. 114–125. DOI: 10.1147/rd.32.0114.

[Rab69] Michael O. Rabin. “Decidability of second-order theories and automata on infinite trees”. In: *Transactions of the American Mathematical Society* 141 (1969), pp. 1–35. URL: http://www.jstor.org/stable/1995086.

[Red12] Roman R. Redziejowski. “An improved construction of deterministic omega-automaton using derivatives”. In: *Fundam. Informaticae* 119.3-4 (2012), pp. 393–406. DOI: 10.3233/FI-2012-744.

[Red99] Roman R. Redziejowski. “Construction of a deterministic-automaton using derivatives”. In: *RAIRO Theor. Informatics Appl.* 33.2 (1999), pp. 133–158. DOI: 10.1051/ITA:1999111.

[Red72] Roman R. Redziejowski. *The theory of general events and its application to parallel programming*. IBM Nordic Laboratory, 1972.

[Saë90] Bertrand Le Saëc. “Saturating right congruences”. In: *RAIRO* 24 (1990), pp. 545–559. DOI: 10.1051/ita/1990240605451.

[SPW91] Bertrand Le Saëc, Jean-Eric Pin, and Pascal Weil. “Semigroups with idempotent stabilizers and applications to automata theory”. In: *Int. J. Algebra Comput.* 1.3 (1991), pp. 291–314. DOI: 10.1142/S0218196791000195.

[Saf88] Schmuel Safra. “On the complexity of ω-automata”. In: *FOCS*. 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948.

[Sch10] Sven Schewe. “Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete”. In: *FSTTCS*. Vol. 8. 2010, pp. 400–411. DOI: 10.4230/LIPIcs.FSTTCS.2010.400.

[Sch20] Sven Schewe. “Minimising good-for-games automata is NP-complete”. In: *FSTTCS*. Vol. 182. 2020, 56:1–56:13. DOI: 10.4230/LIPIcs.FSTTCS.2020.56.

[Sch09] Sven Schewe. “Tighter bounds for the determinisation of Büchi automata”. In: *FOSSACS*. 2009, pp. 167–181. DOI: 10.1007/978-3-642-00596-1_13.

[SV14] Sven Schewe and Thomas Varghese. “Determinising parity automata”. In: *MFCS*. 2014, pp. 486–498. DOI: 10.1007/978-3-662-44522-8_41.

[SV12] Sven Schewe and Thomas Varghese. “Tight bounds for the determinisation and complementation of generalised Büchi automata”. In: *ATVA*. 2012, pp. 42–56. DOI: 10.1007/978-3-642-33386-6_5.

[Sch61a] Marcel Paul Schützenberger. “A remark on finite transducers”. In: *Inf. Control.* 4.2-3 (1961), pp. 185–196. DOI: 10.1016/S0019-9958(61)80006-5.

[Sch61b] Marcel Paul Schützenberger. “On the definition of a family of automata”. In: *Inf. Control.* 4.2-3 (1961), pp. 245–270. DOI: 10.1016/S0019-9958(61)80020-X.

[Sha48] Claude E. Shannon. “A mathematical theory of communication”. In: *Bell Syst. Tech. J.* 27 (1948), pp. 623–656.

[Shi+20] Yingying Shi, Shengping Xiao, Jianwen Li, Jian Guo, and Geguang Pu. “SAT-based automata construction for LTL over finite traces”. In: *APSEC*. IEEE, 2020, pp. 1–10. DOI: 10.1109/APSEC51365.2020.00008.

[Tho90] Wolfgang Thomas. “Automata on infinite objects”. In: *Handbook of theoretical computer science, volume B: formal models and semantics*. Ed. by Jan van Leeuwen. Elsevier and MIT Press, 1990, pp. 133–191. DOI: 10.1016/B978-0-444-88074-1.50009-3.

[Tho97] Wolfgang Thomas. “Languages, automata, and logic”. In: *Handbook of formal languages, volume 3: Beyond words*. 1997, pp. 389–455. DOI: 10.1007/978-3-642-59126-6_7.

[VSL95] Do Long Van, Bertrand Le Saëc, and Igor Litovsky. “Characterizations of rational omega-languages by means of right congruences”. In: *Theor. Comput. Sci.* 143.1 (1995), pp. 1–21. DOI: 10.1016/0304-3975(95)80022-2.

[Var14] Thomas Varghese. “Parity and generalised Büchi automata. Determinisation and complementation”. PhD Thesis. University of Liverpool, 2014.

[Wag79] Klaus Wagner. “On ω-regular sets”. In: *Information and control* 43.2 (1979), pp. 123–177. DOI: 10.1016/S0019-9958(79)90653-3.

[WS21] Thomas Wilke and Sven Schewe. “ω-automata”. In: *Handbook of automata theory*. Ed. by Jean-Éric Pin. European Mathematical Society Publishing House, 2021, pp. 189–234. DOI: 10.4171/Automata-1/6.

[Xia+21] Shengping Xiao, Jianwen Li, Shufang Zhu, Yingying Shi, Geguang Pu, and Moshe Vardi. “On-the-fly synthesis for LTL over finite traces”. In: *AAAI* 35.7 (2021), pp. 6530–6537. DOI: 10.1609/aaai.v35i7.16809.

[Xia+24] Shengping Xiao, Yongkang Li, Xinyue Huang, Yicong Xu, Jianwen Li, Geguang Pu, Ofer Strichman, and Moshe Y. Vardi. “Model-guided synthesis for LTL over finite traces”. In: *VMCAI*. Vol. 14499. 2024, pp. 186–207. DOI: 10.1007/978-3-031-50524-9_9.

[Zie98] Wiesław Zielonka. “Infinite games on finitely coloured graphs with applications to automata on infinite trees”. In: *Theoretical Computer Science* 200.1-2 (1998), pp. 135–183. DOI: 10.1016/S0304-3975(98)00009-7.