Semantic Flowers for Good-for-Games and Deterministic Automata

Daniele Dell'Erba, Sven Schewe, Qiyi Tang, Tansholpan Zhanabekova

[a] *University of Liverpool, Liverpool, UK*

Abstract

We present an innovative approach for capturing the complexity of ω-regular languages using the concept of *flowers*. This semantic tool combines two syntax-based definitions, namely the Mostowski hierarchy of word languages and syntactic flowers. The former is based on deterministic parity automata with a limited number of priorities, while the latter simplifies deterministic parity automata by reducing the number of priorities used, without altering their structure. Synthesising these two approaches yields a semantic concept of flowers, which offers a more effective way of dealing with the complexity of ω-regular languages. This letter provides a comprehensive definition of semantic flowers and shows that it captures the complexity of ω-regular languages. We also show that this natural concept yields simple proofs of the expressive power of good-for-games automata.

Keywords:
Flowers, ω-regular languages, Good-for-Games automata, Acceptance conditions

1. Introduction

In this letter, we define *flowers* that capture the complexity of ω-regular languages.

Flowers are a semantic concept that synergises two syntax based definitions: the Mostowski hierarchy of word languages [1], which is defined over the expressive power of deterministic parity automata with a restricted number of priorities, and syntactic flowers [2], which were introduced to simplify deterministic parity automata (without changing their structure) by reducing the number of priorities used (where possible).

Freeing the beautiful concept of flowers from their syntactic shackles, we derive a semantic definition of flowers that makes no references to the formal acceptors of a language.

We obtain an alternative characterisation of the Mostowski hierarchy that makes no reference to formal acceptors, while having a concept tightly related to syntactic flowers that has proven useful for the simplification of deterministic parity automata, essentially providing a structure preserving translation to an automaton from a minimal level in the Mostowski hierarchy, which coincides with the older Wagner hierarchy [3] that had been defined on Muller automata.

We show that semantic flowers bear the thumbmark of a natural concept: after establishing the standard connection to deterministic parity automata, we show how organically this extends to good-for-games automata—not only nondeterministic, but also alternating ones, and how smoothly it extends to other acceptance mechanisms like Rabin, Streett, and Muller. The link back to Muller automata reiterates the well-known link between the Mostowski [1] and the Wagner hierarchy [3], linking flowers to both of them.

A word automaton recognising a language $\mathcal{L}$ is good-for-games (GFG) if its composition with any game with winning condition $\mathcal{L}$ preserves the game's winner. These automata have been introduced in 2006 by Henzinger and Piterman [4] and are similar in style to good-for-tree automata [5] and history-deterministic automata [6]. GFG automata have been studied intensively (see, for example, [7, 8, 9]).

The one-to-one relationship between deterministic and GFG acceptance condition is easier to establish using semantic flowers than using existing constructions from [10] and does not depend on deep classic insights like the existence of finite state resolvers [11].

2. Preliminaries

We write $\mathbb{N}$ for the set of nonnegative integers. For a finite set X, we use $|X|$ to denote the number of its elements. We write $\mathcal{B}^+(X)$ for the set of positive Boolean formulas over X. We say that a set $Y \subseteq X$ satisfies a formula $\varphi \in \mathcal{B}^+(X)$ if φ evaluates to *true* when all variables in Y are set to *true* and all variables in $X \setminus Y$ are set to *false*. Given a sequence x, $x(i)$ is the i^{th} element of x (i starts from 0).

2.1. Automata over infinite words

An alternating word automaton (with transition-based acceptance condition) is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, \mathsf{Acc})$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, and Acc is an acceptance mechanism. Alternating automata allow to combine both nondeterministic and universal transitions; disjunctions in transition functions model the nondeterministic choices and conjunctions model the universal choices.

An alternating word automaton is nondeterministic if the image of δ consists only of such formulas that contain only disjunctions, and is deterministic if the image of δ is a set of atomic subformulas (states).

A tree is a prefixed-closed set $T \subseteq \mathbb{N}^*$. The elements of T are called nodes and the empty word ε is the root of T. Given a finite alphabet Σ, a Σ-labelled tree is a pair $\langle T, V \rangle$ where T is a tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ.

A run (run tree) of $\mathcal{A}$ on an infinite word w is a Q-labelled tree $\langle T, r \rangle$ such that the following holds:

- $r(\varepsilon) = q_0$;
- Let $x \in T$ with $r(x) = q$ and $\delta(q, w(|x|)) = \varphi$. There is a set $S = \{q_1, \cdots, q_k\} \subseteq Q$ that satisfies φ and for all $1 \leq c \leq k$, we have $x \cdot c \in T$ and $r(x \cdot c) = q_c$.

We represent transition function of nondeterministic automata as $\delta : Q \times \Sigma \to 2^Q$ and call a triple $(q, a, q') \in Q \times \Sigma \times Q$ a transition if $q' \in \delta(q, a)$, and of deterministic automaton as $\delta : Q \times \Sigma \to Q$. A run (path) [1] ρ of an automaton $\mathcal{A}$ on word $w \in \Sigma^\omega$ (resp. $w \in \Sigma^*$) is an infinite (resp. finite) sequence of transitions $\rho(0), \rho(1), \ldots$ where $\rho(i) = (s_i, w(i), s_{i+1})$ is a transition from δ and $s_0 = q_0$. For deterministic automata, every word induces a unique run. Slightly abusing the notation, we write $\delta(q, w) = q'$ if the run of $\mathcal{A}_q$ on the word $w \in \Sigma^*$ ends in q'. We denote by $\mathcal{A}_q$ the automaton obtained from $\mathcal{A}$ by making q the initial state.

The set of transitions that occur infinitely often in a path ρ is denoted by $\mathsf{Inf}(\rho)$. We may drop ρ if it is clear from the context. A run tree is accepting if

[1] In case $\mathcal{A}$ is nondeterministic, on an infinite word w, we have a run tree with only one path.

all paths in the run tree are accepting. We say an infinite word w is accepted by $\mathcal{A}$ if there exists an accepting run tree. We have the following acceptance conditions:

- A parity condition is specified by a priority function $\pi : \delta \to \mathbb{N}$ that assigns a priority to every transition of the automaton. A path ρ is accepted if the highest priority that occurs infinitely often along ρ is even. More formally, when $\max\{\pi(t) \mid t \in \mathsf{Inf}(\rho)\} \equiv_2 0$.

- A Rabin acceptance condition of index k is specified by k pairs of sets of transitions, $\{\langle B_i, G_i\rangle_{1\leq i\leq k}\}$. A path ρ is accepted if there exists a pair of bad and good transitions, $\langle B_i, G_i\rangle$, such that $\mathsf{Inf}(\rho) \cap B_i = \emptyset$ and $\mathsf{Inf}(\rho) \cap G_i \neq \emptyset$. Informally, the accepting path should visit only finitely often transitions of a bad set and infinitely often at least one transition of the corresponding good set.

- A Streett acceptance condition of index k is specified by k pairs of sets of transitions, $\{\langle B_i, G_i\rangle_{1\leq i\leq k}\}$. A path ρ is accepted if, for all pairs of bad and good transitions $\langle B_i, G_i\rangle$, we have $\mathsf{Inf}(\rho) \cap B_i \neq \emptyset$ implies $\mathsf{Inf}(\rho) \cap G_i \neq \emptyset$. Informally, the accepting path should visit infinitely often transitions of a bad set only if it is visiting infinitely often also transitions of the corresponding good set.

- A Muller acceptance condition is specified by a colouring function $\pi : \delta \to \mathbb{N}$ that assigns a colour to every transition of the automaton and a collection of sets of colours $\mathcal{F}$. A path ρ is accepted if the set of colours that occur infinitely often along the path is an element of $\mathcal{F}$. Formally, when $\{\pi(t) \mid t \in \mathsf{Inf}(\rho)\} \in \mathcal{F}$.

There are many ways to define GFG alternating automata. We present the one using the letter games [4, 12]. An alternating automaton $\mathcal{A}$ is GFG if Eve wins her letter game and Adam wins his letter game, respectively.

Eve's letter game proceeds at each turn from a state q of $\mathcal{A}$, starting from the initial state of A, as follows:

- Adam chooses a letter a,

- Adam and Eve play on the one-step arena to resolve the transition $\delta(q, a)$. This can be done by leaving the resolution of disjunctions to

Eve and the resolution of conjunctions to Adam[2].

A play of the letter game thus generates a word w and a path ρ of A on w. Eve wins this play if either $w \notin \mathcal{L}(\mathcal{A})$ or ρ is accepting in A. Adam's letter game is similar, except that Eve chooses letters and Adam wins if either $w \in \mathcal{L}(\mathcal{A})$ or the path ρ is rejecting.

Both players of a GFG alternating automaton have strategies (resolvers) to win their letter games, respectively. Note that strategies can only refer to the past, not the future: these strategies therefore must not depend on the part of the word that is not yet read.

In this paper, we use the common three-letter abbreviations to distinguish types of automata. The first (A, D, N) tells whether the automaton is alternating, deterministic or nondeterministic. The second denotes the acceptance condition (P for parity, R for Rabin, S for Streett, and M for Muller). The last one usually denotes the type of input (i.e W for word or T for tree), in our case we work only with word automata. Therefore, there is no need to specify it and the third letter will be fixed to A for automaton. For example, an DPA is a deterministic parity automaton, and a NRA is a nondeterministic Rabin automaton. We may add G in front of the common three-letter abbreviations for alternating or nondeterministic automata that are GFG, for example, a GAMA is a GFG alternating Muller automaton.

2.2. Syntactic flowers.

Let $\mathcal{A}$ be a deterministic automaton. A *syntactic flower with petals* $c, \dots, d$ in $\mathcal{A}$ consists of

- a reachable state q_c, called the centre of the flower and
- $d - c + 1$ *petals* $\rho_c, \dots, \rho_d$ with the following properties:
    - each petal ρ_i for $c \leq i \leq d$ is a non-trivial run from q_c to itself;

[2]There are different equivalent ways to describe how to find the successor states that boil down to the corner case where the positive Boolean formula is given in disjunctive or conjunctive minimal form: either Eve chooses a satisfying set of successor states first, and then Adam selects one state from this set (which relates to a formula given in disjunctive minimal form) or Adam selects a set of successor states such that each satisfying assignment needs to contain at least one of them, and Eve then selects one state from this set (which relates to a formula given in conjunctive minimal form).

  - for every infinite run $\rho = \rho'_0, \rho'_1, \rho'_2, \dots$ such that
    - ρ'_0 is a finite run from q_0 to q_c, and
    - for all $i > 0$, $\rho'_i \in \{\rho_c, \dots, \rho_d\}$,

    we define the max petal index $e = \max\{i \in \{c, \dots, d\} \mid \forall j \exists k > j.\ \rho'_k = \rho_i\}$. Then ρ is accepting if, and only if, e is even. That is, the parity of the highest index of a petal that occurs infinitely often in ρ determines acceptance.

For parity automata, this could, for example, be the case if the dominating priority for each ρ_i is i.

Syntactic flowers have proven useful for the characterisation of the complexity of parity automata by Niwiński [2]. Moreover, if a parity automaton uses too many priorities (as witnessed by the absence of complex flowers), they can be simplified without changing their structure. (The second citation is for the—simple—extension of this property to transition-based acceptance.)

Theorem 1 (flowers [2, 13]). *Exactly one of the following conditions holds for each ω-regular language $\mathcal{L}$:*

A) L is recognised by a DPA $\mathcal{D}$ that has a (reachable) syntactic flower with petals $c, \dots, d$; or

B) L is recognised by a DPA $\mathcal{D}$ with priorities $c+1, \dots, d+1$. □

3. Flowers

The concept of flowers has so far been pegged to *automata*, hence our reference to them as *syntactic* flowers in the preliminaries.

The core properties of flowers, however, is to characterise the complexity of languages, and this is better characterised on the level of languages. We therefore introduce (semantic) flowers as follows.

Let $\mathcal{L}$ be an ω-regular language over an alphabet Σ. A (semantic) *flower with petals* $c, \dots, d$ in $\mathcal{L}$ consists of

- a finite word $w_s \in \Sigma^*$, called the stem and
- $d-c+1$ *petals* $w_c, \dots, w_d \in \Sigma^+$ with the following properties: for every infinite word $w = w'_0, w'_1, w'_2, \dots$ such that

– $w'_0 = w_s$ is the stem word, and

– for all $i > 0$, $w'_i \in \{w_c, \ldots, w_d\}$.

we define the max petal index $e = \max\{i \in \{c, \ldots, d\} \mid \forall j \exists k > j.\ w'_k = w_i\}$. Then $w \in \mathcal{L}$ if, and only if, e is even. That is, the parity of the highest index of a petal, that occurs infinitely often in w, determines acceptance.

We first show that, for deterministic parity automata, syntactic and semantic flowers are related.

Theorem 2. *Let $\mathcal{L}$ be an ω-regular language recognised by a DPA $\mathcal{A}$. Then $\mathcal{L}$ has a flower with petals $c, \ldots, d$ if, and only if, $\mathcal{A}$ has a syntactic flower with petals $c, \ldots, d$.*

PROOF. First, it is straightforward to construct a semantic flower from a syntactic one: for the stem word w_s, we can take any word w_s such that $\delta(q_0, w_s) = q_c$. Likewise, for each petal ρ_i, we define a word w_i so that ρ_i is the run of $\mathcal{A}_{q_c}$ on w_i.

Such words clearly must exist and trivially satisfy the flower constraints.

The other direction is less straightforward. We can, however, use an indirect proof based on Theorem 1.

We assume for contradiction that $\mathcal{A}$ has no syntactic flower with petals $c, \ldots, d$ and $\mathcal{L}$ has a flower with petals $c, \ldots, d$. Then (by Theorem 1) there is a DPA $\mathcal{D}$ that recognises $\mathcal{L}$ and only uses priorities $c+1, \ldots, d+1$. Towards contradiction, we successively build an infinite run of $\mathcal{D}$ that starts with the stem word w_s, followed by a sequence of petals. The selection of all but the first of these petals (which can be selected arbitrarily; we use petal c) is based on the transition with the maximal priority that occurs in the run sequence that $\mathcal{D}$ produces when traversing the input word from the previous petal: for priority p, we use petal w_{p-1}.

This way, we produce an infinite run of $\mathcal{D}$, whose dominating priority is defined by the dominating priority of the maximal petal index of the input word: if the priority of the run of $\mathcal{D}$ is p, then the maximal petal index is $p - 1$.

Therefore, either $\mathcal{D}$ accepts while the word is not in $\mathcal{L}$, *or* $\mathcal{D}$ rejects and the word is in $\mathcal{L}$.

This provides a contradiction and closes the proof. $\square$

A straightforward consequence is the following corollary.

Corollary 1. *Let $\mathcal{L}$ be an ω-regular language. Then $\mathcal{L}$ is recognised by a DPA with priorities $c+1, \dots, d+1$ if, and only if, $\mathcal{L}$ does not have a flower with petals $c, \dots, d$.* □

Defining flowers semantically proves to be an inroad to a particularly easy extension of characterising the complexity of languages by their Rabin, Streett, or Muller index as well as its extension to GFG automata.

Theorem 3. *Exactly one of the following conditions holds for each ω-regular language $\mathcal{L}$:*

A') L has a semantic flower with petals $c, \dots, d$; or

B') L is recognised by a GAPA (resp. GNPA) $\mathcal{G}$ with priorities $c+1, \dots, d+1$. □

Proof. If A' does not hold, then Corollary 1 says that there is a deterministic parity automaton with these priorities, which is also good for games.

If A' holds, then we can just re-iterate the contradiction argument from the second part of the proof of Theorem 2, this time with a GFG alternating (resp. nondeterministic) automaton $\mathcal{G}$.

Both strategies in letter games, of Adam and Eve, are combined to get a sequence of transitions. This does not change the validity of the contradiction argument: we still either have an accepting path for a word not in $\mathcal{L}$ (in this case, Adam's strategy was not GFG *or* $\mathcal{G}$ accepts this word, and therefore does not recognise $\mathcal{L}$; both cases contradict our assumption) or a rejecting path for a word in $\mathcal{L}$ (in this case, Eve's strategy was not GFG *or* $\mathcal{G}$ rejects this word, and therefore does not recognise $\mathcal{L}$; again both cases contradict our assumption). □

In the case that the automaton is a GNPA, the proof is the same except that we only need Eve's strategy to construct the run (path). This also provides a corollary that deterministic and GFG automata with the same priorities are equally expressive without resorting to complex insights like the existence of finite state resolvers [11].

Corollary 2. *A language $\mathcal{L}$ is recognised by a GAPA (resp. GNPA) with priorities $c, \dots, d$ if, and only if, it is recognised by a DPA with priorities $c, \dots, d$.* □

4. Flowers for Rabin, Streett, and Muller automata

By liberating the flowers from the syntactic structure of acceptors, they can also serve as characterisations of other standard complexity measures of languages: we use them to characterise languages with a Rabin, Streett, and Muller index.

This follows from the observations that parity automata with

- c priorities can be viewed as Muller automata with c colours,
- priorities $1, \ldots, 2k+1$ can be viewed as Rabin automata with k Rabin pairs, and
- priorities $0, \ldots, 2k$ can be viewed as Streett automata with k Streett pairs, respectively

and there is a translation in the other direction through latest appearance records [14, 15, 16] (LARs). The way LARs are updated is deterministic, so that automata keep their type, such as GFG or deterministic.

Theorem 4 ([14, 15, 16]). *An ω-regular language is recognised by a $\star$RA with k Rabin pairs if, and only if, it is recognised by a $\star$PA with priorities $1, \ldots, 2k+1$; a $\star$SA with k Streett pairs if, and only if, it is recognised by a $\star$PA with priorities $0, \ldots, 2k$; and a $\star$MA with k colours if, and only if, it is recognised by a $\star$PA with k priorities, respectively, where $\star \in \{D, GA, GN\}$.*

Together with Corollary 1 and Theorem 2 we now get:

Corollary 3. *An ω-regular language is recognised by a $\star$RA with k Rabin pairs if, and only if, it does not have a flower with petals $0, \ldots, 2k$; a $\star$SA with k Streett pairs if, and only if, it does not have a flower with petals $1, \ldots, 2k+1$; and a $\star$MA with k colours if, and only if, it has no flower with petals $0, \ldots, k-1$* or *no flower with petals $1, \ldots, k$ where $\star \in \{D, GA, GN\}$.*

5. Conclusion

We have introduced semantic flowers as a simple and purely semantic way to define the complexity of ω-regular languages. Taking inspiration from the classic concept of syntactic flowers [2], we retain the 'flower-power' to capture the different priorities needed by a deterministic parity automaton. As a semantic concept, this extends to deterministic parity automata in general.

While this is not surprising, it is useful. For example, the simple proof that a language cannot be captured by an automaton with a simpler acceptance condition now extends, in all its simplicity, to GFG automata. This strengthens the relevance of this natural measure of complexity beyond it being a semantic characterisation of the Mostowski hierarchy of word languages [1].

We also show that, unsurprisingly, the concept can be used to describe the expressive power of Rabin, Streett, and Muller automata, deterministic and GFG, with a restricted number of pairs and colours.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101032464. It has been partially supported by the EPSRC through grant EP/X017796/1 (Below the Branches of Universal Trees).

References

[1] A. W. Mostowski, Regular expressions for infinite trees and a standard form of automata, in: A. Skowron (Ed.), Computation Theory - Fifth Symposium, Zaborów, Poland, December 3-8, 1984, Proceedings, Vol. 208 of Lecture Notes in Computer Science, Springer, 1984, pp. 157–168. doi:10.1007/3-540-16066-3_15.

[2] D. Niwiński, I. 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, Vol. 1373 of Lecture Notes in Computer Science, Springer, 1998, pp. 320–331. doi:10.1007/BFb0028571.

[3] K. Wagner, On ω-regular sets, Information and Control 43 (2) (1979) 123–177. doi:10.1016/S0019-9958(79)90653-3.

[4] T. A. Henzinger, N. Piterman, Solving games without determinization, in: International Workshop on Computer Science Logic, Springer, 2006, pp. 395–410.

[5] O. Kupferman, S. Safra, M. Y. Vardi, Relating word and tree automata, in: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science, IEEE, 1996, pp. 322–332.

[6] T. Colcombet, The theory of stabilisation monoids and regular cost functions, in: International Colloquium on Automata, Languages, and Programming, Springer, 2009, pp. 139–150.

[7] O. Kupferman, Using the past for resolving the future, Frontiers in Computer Science 4 (2023). doi:10.3389/fcomp.2022.1114625.

[8] D. Kuperberg, M. Skrzypczak, On determinisation of good-for-games automata, in: International Colloquium on Automata, Languages, and Programming, Springer, 2015, pp. 299–310.

[9] U. Boker, O. Kupferman, M. Skrzypczak, How deterministic are good-for-games automata?, in: S. V. Lokam, R. Ramanujam (Eds.), 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India, Vol. 93 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 18:1–18:14. doi:10.4230/LIPIcs.FSTTCS.2017.18.

[10] U. Boker, D. Kuperberg, K. Lehtinen, M. Skrzypczak, On the succinctness of alternating parity good-for-games automata, in: N. Saxena, S. Simon (Eds.), 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), Vol. 182 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 41:1–41:13. doi:10.4230/LIPIcs.FSTTCS.2020.41.

[11] J. R. Buchi, L. H. Landweber, Solving sequential conditions by finite-state strategies, Transactions of the American Mathematical Society 138 (1969) 295–311.

[12] U. Boker, K. Lehtinen, Good for games automata: From nondeterminism to alternation, in: W. J. Fokkink, R. van Glabbeek (Eds.), 30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands, Vol. 140 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 19:1–19:16. doi:10.4230/LIPIcs.CONCUR.2019.19.

[13] R. Ehlers, S. Schewe, Natural colors of infinite words, in: A. Dawar, V. Guruswami (Eds.), 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2022, December 18-20, 2022, IIT Madras, Chennai, India, Vol. 250 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 36:1–36:17. doi:10.4230/LIPIcs.FSTTCS.2022.36.

[14] Y. Gurevich, L. Harrington, Trees, automata and games, in: Symposium on Theory of Computing, STOC '82, 1982, p. 60–65.

[15] S. Dziembowski, M. Jurdziński, I. Walukiewicz, How much memory is needed to win infinite games?, in: Symposium on Logic in Computer Science (LICS 97), 1997, pp. 99–110.

[16] J. Kretínský, T. Meggendorfer, C. Waldmann, M. Weininger, Index appearance record for transforming rabin automata into parity automata, in: A. Legay, T. Margaria (Eds.), Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I, Vol. 10205 of Lecture Notes in Computer Science, 2017, pp. 443–460. doi:10.1007/978-3-662-54577-5_26.