TheoretiCS

Positional ω-regular languages

Received Dec 16, 2024
Revised Dec 22, 2025
Accepted Jan 16, 2026
Published Feb 24, 2026

Key words and phrases
Infinite duration games; Positionality; Strategy complexity; Parity automata

Antonio Casares [a]
Pierre Ohlmann [b]

a University of Kaiserslautern-Landau, Germany

b CNRS, LIS, Aix-Marseille University, France

ABSTRACT. In the context of two-player games over graphs, a language L is called positional if, in all games using L as winning objective, the protagonist can play optimally using positional strategies, that is, strategies that do not depend on the history of the play. In this work, we describe the class of parity automata recognising positional languages, providing a complete characterisation of positionality for ω-regular languages. As corollaries, we establish decidability of positionality in polynomial time, finite-to-infinite and 1-to-2-players lifts, and show the closure under union of prefix-independent positional objectives, answering a conjecture by Kopczyński in the ω-regular case.

1. Introduction

1.1 Context: Strategy complexity in infinite duration games

We study games in which two antagonistic players, that we call Eve and Adam, take turns in moving a token along the edges of a given (potentially infinite) edge-coloured directed graph. Vertices of the graph are partitioned between Eve and Adam; when the token reaches a vertex, its owner chooses where to move next. This interaction goes on in a non-terminating mode, producing an infinite path in the graph called a play. The winner of such a play is determined according to a language of infinite sequences of colours W, called the objective of the game; plays producing a sequence of colours in W are winning for Eve, and plays that do not satisfy the objective W are winning for the opponent Adam.

We want to thank Hugo Gimbert, Damian Niwiński and Pierre Vandenhove for interesting discussions on the subject. This work was developed while Antonio Casares was at the University of Warsaw, supported by the Polish National Science Centre (NCN) grant "Polynomial finite state computation" (2022/46/A/ST6/00072). A preliminary version of this article appeared at LICS 2024 [21].

Cite as Antonio Casares, Pierre Ohlmann. Positional ω-regular languages. TheoretiCS, Volume 5 (2026), Article 5, 1-109.

https://theoretics.episciences.org
DOI 10.46298/theoretics.26.5

One of the central applications of games on graphs is the problem of reactive synthesis: given a system interacting with its environment and a formal specification, we aim to design a controller that guarantees the specification is met. The interaction between the system and the environment can be modelled as a game where a winning strategy corresponds to a correct implementation of a controller [14, 54, 4].

In this context, a crucial parameter is the complexity of strategies required by the players to play optimally. Games admitting simple strategies are both easier to solve algorithmically, and the controllers obtained for them can be represented succinctly [14].

Positional strategies. The simplest strategies are positional ones, those that depend only on the current vertex, and not on the history of the play. In this work, we are interested in the following question: Given a fixed objective W, can players always play optimally using positional strategies in all games with winning objective W? If the answer is affirmative for a single player (Eve) we say that W is positional[1]; if it is affirmative for both players, we say that W is bipositional. Also, it might be relevant to consider the question for subclasses of games, in particular, for finite games, or for 1-player games.

Bipositionality. The class of bipositional objectives, both over finite and infinite games, is already well understood. A characterisation of bipositionality over finite games was obtained by Gimbert and Zielonka [32], using two properties called *monotonicity* and *selectivity*. An important and useful corollary of their result is what is commonly known as a *1-to-2-player lift*: an objective W is bipositional over finite games if and only if both players can play optimally using positional strategies in finite 1-player games.

Over infinite games, a very simple and elegant characterisation of bipositionality was given by Colcombet and Niwiński for prefix-independent objectives [27]: a prefix-independent objective W is bipositional if and only if it is the parity objective. In particular, these objectives are necessarily ω-regular. No such characterisation is known for non-prefix-independent objectives (although a generalisation of this result for finite memory without the prefix-independent assumption is studied in [11]).

Positionality. Although positionality is arguably more relevant than bipositionality in the context of reactive synthesis (the controller is built based on Eve's strategies), much less is known for this class. During the 90s, positionality of some central objectives was proved, notably of parity [30] and Rabin languages [33], but the first thorough study of positionality was conducted by Kopczyński in his PhD thesis [34]. There, he provides some sufficient conditions for positionality (which were generalised in [3]) and introduces an important set of conjectures that have greatly influenced research in the area in recent years (see [26, Sect. 6], [10, pg. 2 and

1 Sometimes in the literature the term "half-positional" or "Eve-positional" is used to stress the asymmetric nature of this notion.

ex. 8], [48, pg. 55], [8, 17, 40, 37, 38, 49] for works discussing some of his conjectures). However, no general characterisation was found for positionality.

Recently, Ohlmann made a step forward by characterising positionality via monotone universal graphs [48, 47]. While this characterisation is a valuable tool for proving positionality, it is not constructive and does not directly yield decidability results. Also, Ohlmann's result comes with a caveat: necessity of the existence of universal graphs for positional objectives is only guaranteed for those containing a neutral letter (a letter that does not change membership to W after its removal). He conjectures that this restriction is not essential, as the addition of a neutral letter to any objective should not break positionality.

ω-regular languages. A central class of languages over infinite words is the class of ω-regular languages, which admits several alternative definitions: these are the languages recognised by deterministic parity automata, by non-deterministic Büchi automata, definable using ω-expressions, or using monadic second order logic [13, 45, 46].

One of the main contributions of Kopczyński was to show decidability of positionality over finite games for prefix-independent ω-regular objectives [36, Theorem 2]. His procedure works by enumerating all possible games where positionality might fail (up to a sufficiently large size); it runs in $O(n^{O(n^2)})$ time, where n is the size of a deterministic parity automaton recognising the objective, and does not reveal much about the structure of automata recognising positional languages.

Regarding positionality over arbitrary games and for non-prefix-independent objectives, characterisations have been found for some subclasses of ω-regular objectives. For closed objectives (objectives recognised by safety automata), positionality was characterised by Colcombet, Fijalkow and Horn in 2014 [25].

Recently, a characterisation of positionality for languages recognised by deterministic Büchi automata was provided by Bouyer, Casares, Randour and Vandenhove [8] (see also Proposition 4.17). As a corollary, they establish polynomial-time decidability of positionality for deterministic Büchi automata. However, the conditions they provide are not necessary for positionality in general, for instance, for languages recognised by coBüchi automata.

Finite-to-infinite and 1-to-2-player lifts. As mentioned above, a consequence of Gimbert and Zielonka's result [32] is that, in order to check bipositionality over finite games, it suffices to check whether players can play optimally in 1-player games. Recently, generalisations of 1-to-2-player lifts have been studied in the setting of finite memory by Kozachinskiy [39] and Vandenhove [55, 12, 11]. Vandenhove conjectures that if W is positional over Eve-games (resp. over finite games), then W is positional over all games [55, Conjecture 9.1.1]. This conjecture has been shown to hold in the case of languages recognised by deterministic Büchi automata [8].

Closure under union. One of the recurring themes in Kopczyński's PhD thesis [34] is the following question.

CONJECTURE 1.1 (*Kopczyński's conjecture* [34, Conjecture 7.1]). *Let $W_1, W_2 \subseteq \Sigma^\omega$ be two prefix-independent positional objectives. Then $W_1 \cup W_2$ is positional.*

Very recently, Kozachinskiy [37] disproved this conjecture, but only for positionality over *finite games*. Also, the counter-example he gives is not ω-regular. On the positive side, Kopczyński's conjecture is known to hold in some subclasses of ω-regular objectives: Muller objectives [58], concave objectives [35] and objectives recognised by deterministic Büchi automata [8], as well as for the family of Σ_2^0 objectives (objectives recognised by infinite coBüchi automata) [49]. Kopczyński's conjecture and this latter result have been generalised to the setting of finite memory [20, Section 6.3]. Solving Kopczyński's conjecture over infinite games is one of the driving open questions for the field.

1.2 Contributions and organisation

Our main contribution is a characterisation of positionality for ω-regular languages, stated in Theorem 3.1. We propose a syntactic description of a family of deterministic parity automata, so that any automaton in this class recognises a positional language, and any positional language can be recognised by such an automaton. In fact, we describe two slightly different such families, called, respectively, fully progress consistent signature automata and ε-completable automata. These families offer distinct advantages and complement our intuitions on positionality.

From this characterisation, we derive multiple corollaries that address the majority of open questions related to positionality in the case of ω-regular languages:

1. **Decidability in polynomial time.** Given a deterministic parity automaton $\mathcal{A}$, we can decide in polynomial time whether $\mathcal{L}(\mathcal{A})$ is positional or not (Theorem 3.2).
2. **Finite-to-infinite and 1-to-2-players lift.** An ω-regular objective W is positional over arbitrary games if and only if it is positional over finite, ε-free Eve-games (Theorem 3.3). This answers a question raised by Vandenhove [55, Conjecture 9.1.1].
3. **Closure under union.** The union of two ω-regular positional objectives is positional, provided that one of them is prefix-independent (Theorem 3.4). This solves a stronger variant of Kopczyński's conjecture in the case of ω-regular languages.
4. **Closure under addition of a neutral letter.** If W is ω-regular and positional, the objective obtained by adding a neutral letter to W is positional too (Theorem 3.9). This solves Ohlmann's conjecture in the case of ω-regular languages.

We obtain some additional results for classes of objectives that are not necessarily ω-regular. We relax the ω-regularity hypothesis in two orthogonal ways.

5. **Characterisation of bipositionality of all objectives.** We extend the characterisation of bipositionality of Colcombet and Niwiński [27] to all objectives, getting rid of the prefix-independence assumption (Theorem 7.1).
6. **Characterisation of positionality of closed and open objectives.** We characterise positionality for closed and open objectives. We also obtain as corollaries 1-to-2 players lifts and closure under addition of a neutral letter for these classes of objectives.

Technical tools

We would like to highlight some technical tools that take primary importance in our proofs.

Universal graphs. In general, showing that a given objective is positional can be challenging, as we need to show that *for every game* Eve can play optimally using positional strategies. Ohlmann's characterisation using monotone universal graphs provides a clear path to prove positionality (see Proposition 2.2). We rely on this result to show that parity automata satisfying the syntactic conditions imposed in Theorem 3.1 do indeed recognise positional languages.

History-deterministic automata. History-deterministic automata are a model in between deterministic and non-deterministic ones; we refer to [7, 42] for detailed expositions on them. Although the statements of our results do not mention history-determinism, they appear naturally in two different parts of our proofs:

— Establishing necessity of the syntactic conditions from our main characterisation requires a very fine control of the structure of automata. We develop a technique for decomposing automata, for which we need to use and generalise the methods introduced by Abu Radi and Kupferman [1] for the minimisation of HD coBüchi automata.

— To prove that these conditions are sufficient, we construct a monotone universal graph from a signature automaton. To facilitate this process, we first "saturate" automata, adding as many transitions as possible without modifying the languages they recognise. This procedure generates non-determinism, but preserves history-determinism, the key property that allows us to prove universality of the obtained graph.

We believe that this use of history-determinism showcases their usefulness and canonicity.

Normal form of parity automata. In our central proof, we rely on a normal form of parity automata, as defined in [19, Section 6.2]. Automata in normal form present a set of properties that simplify manipulating them and reasoning about their runs. We make consistent use of these properties in our combinatorial arguments. This normal form is commonly used in the literature and applied in areas such as the study of history-deterministic coBüchi automata [41, 1, 29] or automata learning [5].

Congruences for parity automata. Since the beginning of the theory of finite automata, the notion of congruence has played a fundamental role [2, 51, 43]. Here, we propose a notion of congruences for parity automata that make it possible to build quotient automata that are compatible with the acceptance condition. This newly introduced vocabulary allows us to formalise the details of the proof of Theorem 3.1 in a simpler way. We believe that it will be useful for the study of parity automata in other contexts.

Organisation of the paper

After introducing some general definitions and terminology used throughout the paper, we begin Section 3 by stating the characterisation result (Theorem 3.1) and its main consequences, without providing formal details about the technical concepts appearing in its statement. Section 4 is a warm-up for the definitions used in the main characterisation and for the techniques used in its proof. We gradually introduce conditions that are necessary for positionality, obtaining partial results and providing numerous examples along the way. Section 5 contains the most technical part of the paper. We introduce the notions of signature automata, full progress consistency and ε-complete automata appearing in the statement of Theorem 3.1, and we give a proof of it. Nevertheless, most details in the proof of necessity are relegated to Appendix A. In Section 6 we provide two conceptually different polynomial-time decision methods for deciding positionality. Sections 7 and 8 contain, respectively, the two last contributions of the paper: a characterisation of bipositionality for all objectives and a characterisation of positionality for open and closed objectives.

2. Preliminaries

We introduce definitions and notations used throughout the paper. First we introduce in Section 2.1 games and positionality, as well as the more technical notion of universal graphs used in our proofs for positionality. In Section 2.2 we introduce definitions about parity automata and notions about congruences for them.

The reader who does not plan to get into the more technical details of Sections 5 and 6 can skip Subsections 2.1.3 and 2.2.2 from this preliminaries. Also, the hyperlinks on words should help the reader to easily refer to the definitions.

2.1 Games and positionality

2.1.1 Games on graphs

Σ-graphs. A Σ-*graph* G is given by a (potentially infinite) set of vertices V together with a set of coloured directed edges $E \subseteq V \times \Sigma \times V$. We write $v \xrightarrow{c} v'$ to refer to an edge in G with source v, target v', and colour c. This notation naturally extends to finite and infinite paths. The size of a graph G is defined to be the cardinality of V.

GLOBAL HYPOTHESIS. We assume throughout the paper that Σ-graphs do not contain *dead-ends*, that is, every vertex has at least one outgoing edge. (This assumption is useful when considering infinite paths.)

Games. A *game* is an edge-coloured graph together with a set of winning sequences of colours and a partition of the vertices into those controlled by a player named *Eve* and her opponent, named *Adam*. Formally, it is represented by a tuple $\mathcal{G} = (V, E, V_{\mathrm{Eve}}, W)$, where $G = (V, E)$ is a $\Sigma \cup \{\varepsilon\}$-graph (called the *game graph*), V_{Eve} is the set of vertices owned by Eve and $W \subseteq \Sigma^\omega$ is the *winning objective*. Letter $\varepsilon \notin \Sigma$ is an additional letter used to represent *uncoloured edges*; we impose that no infinite path in G is composed exclusively of ε-edges. Games not containing uncoloured edges are called *ε-free*. We let $V_{\mathrm{Adam}} = V \setminus V_{\mathrm{Eve}}$ be the vertices controlled by Adam. An *Eve-game* is a game $\mathcal{G}$ in which all the vertices are controlled by Eve, that is, $V = V_{\mathrm{Eve}}$. A game having W as winning objective is called a *W-game*.

Unless stated otherwise, we take the point of view of player Eve; expressions as "winning" will implicitly stand for "winning for Eve", and strategies will be defined for her.

In this paper, the words "language" and "*objective*" are synonyms.

Plays. In a game, players move a pebble from one vertex to another for an infinite amount of time. The player who owns the vertex v where the pebble is placed chooses an edge $v \xrightarrow{c} v'$ and the pebble travels through this edge to its target, producing colour c. In this way, they produce a path $\rho = v_0 \xrightarrow{c_0} v_1 \xrightarrow{c_1} v_2 \xrightarrow{c_2} \cdots \in E^\omega$, that we call a *play*. Such a play is *winning* (for Eve) if the sequence $w \in \Sigma^\omega$ obtained by removing from $c_0c_1c_2\dots$ the occurrences of ε belongs to W. We say that it is *losing* (or *winning for Adam*) on the contrary.

We let $\mathsf{Paths}(\mathcal{G})$ be the set of finite paths in $\mathcal{G}$; these are either non-empty sequences in E^+ or a vertex $v \in V$ (encoding the empty path starting in that vertex).

Strategies and winning regions. A *strategy* (for Eve) is a function $\mathsf{strat} : \mathsf{Paths}(\mathcal{G}) \to E$, that tells Eve which move to choose after any possible finite play. We say that a play $\rho \in E^\omega$ is *consistent with the strategy* strat if after each finite prefix ρ' of ρ ending in a vertex controlled by Eve, the next edge in ρ is $\mathsf{strat}(\rho')$. We say that the strategy strat is *winning from* a vertex $v \in V$ if all infinite plays starting in v consistent with strat are winning. If such a strategy exists, we say that Eve *wins $\mathcal{G}$ from v*. Strategies for Adam are defined symmetrically.

The *winning region* of a game $\mathcal{G}$, written $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G})$, is the set of vertices $v \in V$ such that Eve wins $\mathcal{G}$ from v. We say that a strategy is *optimal (for Eve)* if it is winning from all vertices in $\mathcal{W}in_{\mathrm{Eve}}(\mathcal{G})$. (Note that Eve always has an optimal strategy.)

Determinacy. We say that a game $\mathcal{G}$ is *determined* if either Eve or Adam have a winning strategy from v, for every vertex v. In this work, all games will be determined, as by Martin's theorem [44] games using Borel objectives are determined, and all objectives that we will consider (for instance, all ω-regular objectives) are Borel.

Graphical representation of games. We use circles to represent vertices controlled by Eve and squares to represent those controlled by Adam. We will allow ourselves to consider games with edges labelled by finite words $w = w_1 w_2 \dots w_n \in \Sigma^*$. Formally, such transitions will stand for a sequence of n transitions, with $n-1$ intermediate vertices. We represent this kind of transitions by a wiggly arrow. We will also use this notation for infinite words: for $w \in \Sigma^\omega$ we write $v \overset{w}{\rightsquigarrow}$ for an infinite sequence of edges labelled with the letters of w starting from v. In this case, the resulting game graph is necessarily infinite.

2.1.2 Positionality

Positional strategies. We say that a strategy $\mathsf{strat} : \mathsf{Paths}(\mathcal{G}) \to E$ is *positional* if there exists a mapping $\sigma \colon V_{\mathrm{Eve}} \to E$ such that for every finite play $\rho = v_0 \xrightarrow{c_0} \dots \xrightarrow{c_{n-1}} v_n$ ending in a vertex v_n controlled by Eve we have:

$$\mathsf{strat}(\rho) = \sigma(v_n).$$

That is, a strategy is positional if the choice of the next transition only depends on the current position, and not on the history of the path.

We say that Eve (resp. Adam) can *win positionally* from a subset $A \subseteq V$ if there is a positional strategy that is winning from any vertex in A. We say that Eve (resp. Adam) can *play optimally in $\mathcal{G}$ using a positional strategy* if she can win positionally from her winning region.

Positional objectives. An objective $W \subseteq \Sigma^\omega$ is *positional* if for every W-game, Eve can play optimally using positional strategies.[2] We say that W is *bipositional* if both W and $\Sigma^\omega \setminus W$ are positional, or, equivalently, if both Eve and Adam can play optimally using positional strategies in W-games. If $\mathcal{X}$ is a subclass of W-games (notably, finite, ε-free and Eve-games), we say that W is *positional over $\mathcal{X}$ games* if for every W-game in $\mathcal{X}$, Eve can play optimally using positional strategies. The same terminology is used for bipositionality.

REMARK 2.1. Our notion of positionality uses what sometimes are called *uniform strategies*, that is, we require that a single positional strategy suffices to win independently of the initial vertex. This notion is strictly stronger than the non-uniform version in which we allow to use different strategies depending on the initial vertex. Said differently, the fact that Eve always has optimal strategies does not hold if we restrict to positional strategies, see Figure 1 for an example. Nevertheless, we note that if ε-edges are allowed, or for prefix-independent objectives, both notions of positionality coincide, as we can always add a vertex controlled by Adam from which he picks the starting position.

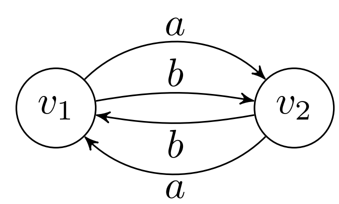

Figure 1. Consider the game above, where Eve controls both vertices v_1 and v_2. Let $W = ab(a+b)^\omega$ be the winning condition of the game, that is, Eve wins if the play starts by ab. She has two positional strategies strat_1 and strat_2 winning from v_1 and v_2, respectively. However, no positional strategy is winning from the entire winning region $\{v_1, v_2\}$.

2.1.3 Universal graphs

We now introduce universal graphs, which will serve as our main tool for deriving positionality results.

Morphisms of Σ-graphs. Given two Σ-graphs $G = (V, E)$ and $G' = (V', E')$, a *morphism of Σ-graphs* ϕ from G to G' is a map $\phi : V \to V'$ such that for each edge $v \xrightarrow{c} v'$ in G, it holds that $\phi(v) \xrightarrow{c} \phi(v')$ defines an edge in G'. We write $\phi : G \to G'$ to denote that ϕ is a morphism.

Universality. Given a Σ-graph G, a vertex v of G and an objective $W \subseteq \Sigma^\omega$, we say that v *satisfies* W in G if for any infinite path $v \overset{w}{\rightsquigarrow}$ in G, it holds that $w \in W$. Given a cardinal κ, a graph U is (κ, W)-*universal* if all graphs G of size $< \kappa$ admit a morphism $\phi : G \to U$ such that any vertex v that satisfies W in G is mapped to a vertex $\phi(v)$ that satisfies W in U.

Monotonicity. A *totally ordered graph* (resp. *well-ordered graph*) is a graph G together with a total order (resp. well-order) $\leq$ on its vertex set V. Such a graph is called *monotone* if

$$u \leq v,\ v' \leq u' \text{ and } u \xrightarrow{c} u' \text{ in } G \implies v \xrightarrow{c} v' \text{ in } G.$$

We often note the conditions on the left by $v \geq u \xrightarrow{c} u' \geq v'$.

We now state our main tool for proving positionality.

PROPOSITION 2.2 ([47, Theorem 3.1]). *Let $W \subseteq \Sigma^\omega$ be an objective. If for all cardinals κ there exists a (κ, W)-universal well-ordered monotone graph, then W is positional over all games.*

Universality for trees. A Σ-*tree* is a Σ-graph T with a distinguished vertex t_0, called the *root*, and such that every vertex t of T admits a unique path from the root. Since graphs (and in particular trees) are assumed without dead-ends, trees are always infinite. We say that a tree T *satisfies* W if its root t_0 satisfies W in T.

2 As in other definitions, the notion of positionality depends not only on the set W, but also on the set of colours Σ. As the set of colours will always be clear from the context, we omit Σ in the notations.

We say that a graph U is (κ, W)-*universal for trees* if all trees T of size $< \kappa$ which satisfy W admit a morphism $\phi\colon T \to U$ mapping the root t_0 to a vertex $\phi(t_0)$ that satisfies W in U.

Given an ordered Σ-graph U, we let $U^\top$ be the Σ-graph obtained by adding a fresh vertex $\top$, maximal for the order of the graph, with transitions $\top \xrightarrow{a} v$ for every $a \in \Sigma$ and every vertex v of the graph. The following useful result follows directly from the proof of [47, Theorem 3.1] (see also [20, Theorem 3.1]).

LEMMA 2.3. *Let $W \subseteq \Sigma^\omega$ be an objective and κ a cardinal. If U is a well-ordered monotone graph that is (κ, W)-universal for trees, then $U^\top$ is well-ordered monotone (κ, W)-universal (for graphs).*

Therefore, thanks to Proposition 2.2, building graphs that are universal for trees suffices to prove positionality.

Universal graph for the parity objective. As an important example, we give a universal graph for the parity objective; it is implicit in the works of Emerson and Jutla [30] and Walukiewicz [56]. In the latter, the term *signatures* was used to name tuples of ordinals ordered lexicographically (term first used in [52]). Such a tuple is meant to count, for each odd priority, how many times it is seen before a stronger (even or odd) priority.

EXAMPLE 2.4 (Universal graph for the parity objective). Consider the *parity objective* over $[0, d]$, (we assume d even, and use min-parity):

$$\mathsf{parity} = \{w \in \{0, \dots, d\}^\omega \mid \liminf w \text{ is even}\}.$$

Fix a cardinal κ. We define a graph U_{parity} having as set of vertices tuples $(\lambda_1, \lambda_3, \dots, \lambda_{d-1}) \in \kappa^{d/2}$ that we consider ordered lexicographically. This is indeed a well-order. We let its edges be:

$$(\lambda_1, \dots, \lambda_{d-1}) \xrightarrow{x} (\lambda'_1, \dots, \lambda'_{d-1}) \iff \begin{cases} (\lambda'_1, \dots, \lambda'_{x-1}) \leq (\lambda_1, \dots, \lambda_{x-1}) & \text{if } x \text{ is even,} \\ (\lambda'_1, \dots, \lambda'_x) < (\lambda_1, \dots, \lambda_x) & \text{otherwise.} \end{cases}$$

Where the order between truncated tuples as on the right is also the lexicographic one. A representation of the graph U_{parity} appears in Figure 2.

Clearly, U_{parity} is monotone. We show in Lemma 2.5 below that all vertices in U_{parity} satisfy parity. Lemma 2.6 states that U_{parity} is $(\kappa, \mathsf{parity})$-universal for trees, so, by Lemma 2.3, $U^\top_{\mathsf{parity}}$ is a well-ordered monotone $(\kappa, \mathsf{parity})$-universal graph. ◆

LEMMA 2.5. *Every infinite path in U_{parity} satisfies the objective* parity.

PROOF. Consider an infinite path $\rho = (\lambda^1_1, \dots, \lambda^1_{d+1}) \xrightarrow{w_1} (\lambda^2_1, \dots, \lambda^2_{d+1}) \xrightarrow{w_2} \dots$ in U_{parity}. Let x be the minimal priority appearing infinitely often in ρ, which we assume odd for contradiction.

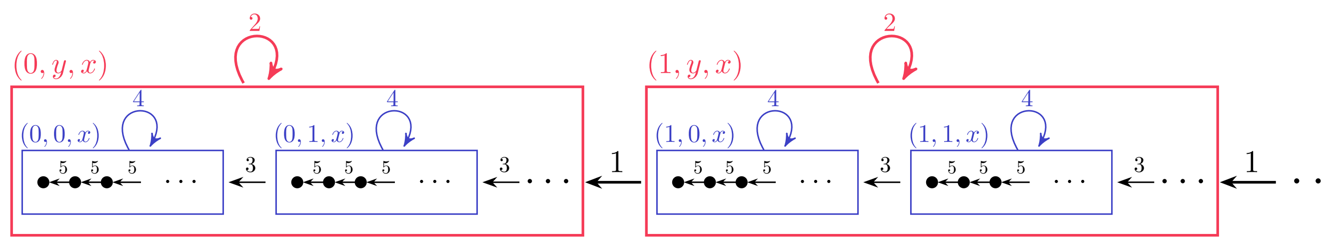

Figure 2. Universal graph $\mathcal{U}_{\mathrm{parity}}$ for the parity objective over priorities $[0, 5]$. Vertices are ordered from left to right. Edges between two boxes $B_1 \xrightarrow{x} B_2$ represent that there are edges $v_1 \xrightarrow{x} v_2$ for all $v_1 \in B_1$ and all $v_2 \in B_2$. Edges obtained by monotonicity are not all represented: if $v \xrightarrow{x} v'$ and $v'' \le v'$, then $v \xrightarrow{x} v''$ too; for example, by reading colour 5 from a vertex v one can go to any vertex strictly on the left of v. Edges coloured 0 are not depicted in the figure: they appear between every pair of vertices. The label of a box represents the forms of the names of vertices inside it.

Then, from some position, no priority $< x$ is read, thus the sequence of prefixes $(\lambda_1^i, \ldots, \lambda_x^i)$ is decreasing and moreover strictly decreases in infinitely many places. This contradicts well-foundedness of the lexicographical order over tuples of ordinals. ■

LEMMA 2.6. *The graph U_{parity} is $(\kappa, \mathrm{parity})$-universal for trees.*

PROOF. Take a tree T of size $< \kappa$ which satisfies parity; note that by prefix-independence, all vertices in T satisfy parity. We aim to construct a morphism $\phi : T \longrightarrow U_{\mathrm{parity}}$.

Fix a vertex $t \in T$ and an odd priority $y \in \{1, 3, \ldots, d-1\}$. Then, in any path $t \xrightarrow{w}$ in T there are only finitely many occurrences of y before a smaller priority appears. We may define an ordinal $\mathrm{rank}_y(t)$ capturing the number of such occurrences: $\mathrm{rank}_y(t)$ satisfies that, if t' is an x-sucessor of t (that is, $t \xrightarrow{x} t'$), then:

- $\mathrm{rank}_y(t') \le \mathrm{rank}_y(t)$, if $y < x$, and
- $\mathrm{rank}_y(t') < \mathrm{rank}_y(t)$ if $x = y$.

It can be verified that $\phi : v \mapsto (\mathrm{rank}_1(v), \mathrm{rank}_3(v), \ldots, \mathrm{rank}_{d-1}(v))$ defines a morphism from T to U_{parity}. ■

2.2 Automata over infinite words

2.2.1 Parity automata

A *(non-deterministic) parity automaton* over the alphabet Σ is represented by a tuple $\mathcal{A} = (Q, \Sigma, q_{\mathrm{init}}, \Delta, \mathrm{p})$, where Q is a finite set of states, Σ is a set of letters called the input alphabet (possibly infinite), q_{init} is the *initial state*, $\Delta \subseteq Q \times \Sigma \times Q$ is a set of transitions, and $\mathrm{p}\colon \Delta \to [d_{\min}, d_{\max}]$ where $[d_{\min}, d_{\max}] \subseteq \mathbb{N}$ is a finite subset of numbers that we refer to as *priorities*. We write $q \xrightarrow{a:x} q'$ to indicate that there is a transition $e = (q, a, q') \in \Delta$ with $\mathrm{p}(e) = x$. We refer to transitions of a parity automaton labelled with priority $x \in \mathbb{N}$ as *x-transitions*. Similarly, we refer to transitions having input letter $a \in \Sigma$ as *a-transitions*. The difference between the two uses of the term should be clear from the context. An automaton $\mathcal{A}' = (Q', \Sigma', q'_{\mathsf{init}}, \Delta', \mathsf{p}')$ is a *subautomaton* of $\mathcal{A}$ if $Q' \subseteq Q$, $\Delta' \subseteq \Delta$ and p' is the restriction of p to Δ'.

For a state $q \in Q$, we write $\mathcal{A}_q$ for the automaton obtained by setting q as initial state.

We assume in all the paper that all automata are *complete*, that is if for every $q \in Q$ and $a \in \Sigma$, there is at least one transition $q \xrightarrow{a:s}$.

A *strongly connected component* (shortened SCC) of an automaton $\mathcal{A}$ is a maximal set of states $S \subseteq Q$ such that any pair of states in S are interreachable. We say that a SCC is *trivial* if it is a singleton. A state q is *recurrent* if it belongs to some non-trivial SCC, and *transient* otherwise.

An *automaton structure* $\mathcal{S}$ is an automaton without a colouring function p, and $\mathcal{A}$ is a *parity automaton on top of* $\mathcal{S}$ if it has been obtained by defining a colouring p on $\mathcal{S}$.

Runs and recognisability. A *run over* an infinite word $w = a_0a_1a_2\cdots \in \Sigma^\omega$ in $\mathcal{A}$ is a path

$$\rho = q_{\mathsf{init}} \xrightarrow{a_0:x_0} q_1 \xrightarrow{a_1:x_1} q_2 \xrightarrow{a_2:x_2} \cdots \in \Delta^\omega.$$

It is *accepting* if

$$\min\{x \in \mathbb{N} \mid x = x_i \text{ for infinitely many } i\} \text{ is even,}$$

and *rejecting* if the minimal priority produced infinitely often is odd (note that we use the min-parity condition). A word $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if there exists an accepting run over w. The *language recognised* by an automaton $\mathcal{A}$ is the set

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\omega \mid w \text{ is accepted by } \mathcal{A}\}.$$

Two automata recognising the same language are said to be *equivalent*. A language is called *ω-regular* if it can be recognised by a parity automaton.

For an even priority x, we say that a word $w \in \Sigma^\omega$ can be *accepted with priority* x in $\mathcal{A}$ if there exists a run over w such that the minimal priority produced infinitely often is x. For an odd priority x, we say that w *is rejected with priority* x if in every run over w the minimal priority produced infinitely often is x. Note that in a non-deterministic automaton, not all rejected words need to have a well-defined rejecting priority.

REMARK 2.7 (Transition-based acceptance). We emphasise that in our definition, the acceptance condition is put over the *transitions* of the automaton. This will be a crucial element in our characterisation. We refer to [18, Chapter VI] for further discussions on the comparison between transition-based and state-based automata.

Büchi and coBüchi automata. A *Büchi automaton* is a parity automaton using $[0, 1]$ as its set of priorities. Parity automata using $[1, 2]$ as set of priorities are called *coBüchi*. We say that a language W is *Büchi recognisable* (resp. *coBüchi recognisable*) if it can be recognised by a deterministic Büchi automaton (resp. deterministic coBüchi automaton). We note that these classes are incomparable and strict subclasses of the ω-regular languages.

For $u \in \Sigma^*$, we write $\mathtt{Inf}(u)$ (resp. $\mathtt{Fin}(u)$) to denote the language of infinite words containing infinitely often (resp. finitely often) the factor u. We note that these languages are Büchi and coBüchi recognisable, respectively. We also write $\mathtt{No}(u)$ for the language of infinite words avoiding any occurrence of the factor u.

Determinism and homogeneity. We say that an automaton $\mathcal{A}$ is *deterministic* if for every $q \in Q$ and $a \in \Sigma$, there is only one a-transition $q \xrightarrow{a:x}$ outgoing from q. Let $\Delta' \subseteq \Delta$ be a subset of transitions of an automaton $\mathcal{A}$. We say that $\mathcal{A}$ is *deterministic over* Δ' if the restriction of $\mathcal{A}$ to Δ' is deterministic, that is, for each letter a, there is a most one outgoing a-transition in Δ' from each state. Any parity automaton admits an equivalent deterministic one [46].

We say that a parity automaton $\mathcal{A}$ is *homogeneous* if for every state $q \in Q$ and letter $a \in \Sigma$, if $q \xrightarrow{a:x} p$ is a transition in $\mathcal{A}$, then any other a-transition from q produces priority x.

REMARK 2.8. Let $\mathcal{A}$ be a homogeneous parity automaton that is deterministic over transitions producing priority x. If $q \xrightarrow{a:x} p$ is a transition in $\mathcal{A}$, then there is no other outgoing a-transition from q.

Notations for paths. For two states q, p of a parity automaton $\mathcal{A}$ and a finite word $w \in \Sigma^*$, we write $q \overset{w:x}{\rightsquigarrow} p$ if there exists a path from q to p labelled w such that the minimal priority appearing on it is $x \in \mathbb{N}$. We write $q \overset{w:\geq x}{\rightsquigarrow} p$ (resp. $q \overset{w:>x}{\rightsquigarrow} p$) to denote that there exists such a path producing no priority $< x$ (resp. $\leq x$). This is possibly the empty path $q \xrightarrow{\varepsilon} q$, producing no priority. We use similar notations for $\leq x$ and $< x$. We generalise these notations for infinite paths: for an infinite word $w \in \Sigma^\omega$ we write $q \overset{w:x}{\rightsquigarrow}$ if there exists an infinite path from q labelled w such that the minimal priority seen on it is x.

We may apply this notations to non-deterministic automata – hence the use of an existential quantification – though in most cases we will work with deterministic ones.

History-deterministic automata. Let $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \Delta, \mathsf{p})$ be a (non-deterministic) parity automaton. A *resolver* for $\mathcal{A}$ is a function $\mathsf{r} \colon \Delta^* \times \Sigma \to \Delta$ such that, for all words $w = a_0 a_1 \cdots \in \Sigma^\omega$, the sequence $e_0 e_1 \cdots \in \Delta^\omega$, called the *run induced by* r over w and defined by $e_i = \mathsf{r}(e_0 \dots e_{i-1}, a_i)$, is actually a run over w in $\mathcal{A}$. We write $q_{\mathsf{init}} \underset{\mathsf{r}}{\overset{w:x}{\rightsquigarrow}} q$ to denote that the run induced by r over w produces x as minimal priority and lands in q.

We say that the resolver is *sound* if it satisfies that, for every $w \in \mathcal{L}(\mathcal{A})$, the run induced by r over w is an accepting run. In other words, r should be able to construct an accepting run in $\mathcal{A}$ letter-by-letter with only the knowledge of the word so far, for all words in $\mathcal{L}(\mathcal{A})$.

An automaton $\mathcal{A}$ is called *history-deterministic* (shortened HD) if there is a sound resolver for it. History-deterministic automata are sometimes called good-for-games in the literature, we refer to [7] for a discussion on the relation between these notions and a survey on them.

Normal form of parity automata Let $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \Delta, \mathsf{p})$ be a parity automaton. We say that a labelling $\mathsf{p}' \colon \Delta \to [d'_{\min}, d'_{\max}]$ is *equivalent to* p over $\mathcal{A}$ if for every cycle $\ell \subseteq \Delta$, $\min \mathsf{p}(\ell)$ is even if and only if $\min \mathsf{p}'(\ell)$ is even. We say that a SCC of a parity automaton is *positive* if the minimal priority appearing on it is even, and *negative* otherwise.

DEFINITION 2.9 (Normal form [19]). A parity automaton $\mathcal{A}$ is in *normal form* if it holds that for every pair of states q, p in a same positive SCC (resp. negative SCC), whenever there is a path $q \xrightarrow{w:x} p$ producing x as minimal priority, then, for every $0 \leq y \leq x$ (resp. $1 \leq y \leq x$), there is a returning path $p \xrightarrow{w':y} q$ producing y as minimal priority.[3]

That is, if $\mathcal{A}$ is in normal form, the restriction of $\mathcal{A}$ to priorities $\geq x$ consists in a disjoint union of strongly connected components. Moreover, if priority $y > x$ appears in one of these SCCs, then all priorities between x and y appear on it.

Every parity automaton admits an equivalent labelling so that the obtained automaton is in normal form, and this labelling is unique (except for the labelling of edges changing of SCC, for which no condition is imposed). This labelling is the one assigning to each transition the smallest possible priority [19, Theorem 6.27]. Moreover, it can be computed in polynomial time. We refer to this process as the *normalisation*.

PROPOSITION 2.10 ([16]). *Given a parity automaton $\mathcal{A}$, we can compute in polynomial time an equivalent labelling defining an automaton in normal form.*

Automata with ε-transitions. An *automaton with ε-transitions* is defined just as an automaton over the alphabet $\Sigma \cup \{\varepsilon\}$, where $\varepsilon \notin \Sigma$ is a distinguished letter. The language of an automaton $\mathcal{A}$ with ε-transitions is the set of words $w \in \Sigma^\omega$ such that there exists $w' \in (\Sigma \cup \{\varepsilon\})^\omega$ which is accepted by $\mathcal{A}$ and such that w is obtained from w' by removing all occurrences of the letter ε.

Alphabets of words. As an important element of our main proof, we will need to consider automata whose transitions are labelled from an alphabet $A \subseteq \Sigma^+$ of finite words. Such an automaton defines a language $L \subseteq A^\omega$ which we would like to see as a language $L \subseteq \Sigma^\omega$; however, this may pose a problem if a word $w \in \Sigma^\omega$ admits several decompositions in A^ω.

We say that a set $A \subseteq \Sigma^+$ is a *uniquely decodable alphabet* if any word $w \in \Sigma^\omega$ admits a unique decomposition as elements of A: for any infinite sequences $a_1, a_2, \dots$ and $a'_1, a'_2, \dots$ of elements of A, if $a_1 a_2 \cdots = a'_1 a'_2 \dots$ then $a_i = a'_i$ for all i.

3 This notion can be refined in the natural way to fit automata not using priority 0 at all (for instance coBüchi automata). We refer to [19] for formal details.

We will only consider alphabets of words $A \subseteq \Sigma^+$ which are *prefix codes*: if $a \in A$, then no proper prefix of a belongs to A. It is an easy check that these are uniquely decodable, and therefore one may indeed see a language $L \subseteq A^\omega$ as $L \subseteq \Sigma^\omega$.

2.2.2 Congruences and monotone preorders over automata

Equivalence relations and preorders. We will use $\sim_X$ to denote different equivalence relations, and $[q]_X$ to denote the *equivalence class* of an element q (which is usually a state in an automaton).

A *preorder* $\sqsubseteq_X$ is a binary relation that is reflexive and transitive. (We reserve the symbol $\sqsubseteq$ for preorders over states of automata.) The *equivalence relation induced from a preorder* $\sqsubseteq_X$ is the relation defined as:

$$q \sim_X q' \iff q \sqsubseteq_X q' \text{ and } q' \sqsubseteq_X q.$$

Given a preorder $\sqsubseteq_X$, we always write $\sim_X$ for the induced equivalence relation, and simply write $\sqsubseteq$ for the induced order over equivalence classes, for instance we may write $[q]_X \sqsubseteq [q']_X$. A preorder is *total* if every pair of elements are comparable.

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two binary relations over a set A (usually preorders or equivalence relations). We say that $\mathcal{R}_1$ is a *refinement* of $\mathcal{R}_2$ if for all $q, p \in A$, $q \mathcal{R}_1 p$ implies $q \mathcal{R}_2 p$. We note that if $\sqsubseteq_1$ is a preorder refining $\sqsubseteq_2$, then the induced equivalence relation $\sim_1$ refines $\sim_2$.

Congruences, uniformity and monotonicity. Let $\mathcal{A}$ be a (possibly non-deterministic) automaton over Σ with states Q and transitions Δ. Let $\sim$ be an equivalence relation over Q and let $\Delta' \subseteq \Delta$ be a subset of transitions (usually Δ' will be the set of transitions using a given priority). We say that transitions of Δ' *are uniform over* $\sim$*-classes* if for all $q \sim q'$ and $a \in \Sigma$, if $q \xrightarrow{a} p \in \Delta'$ then all a-transitions $q' \xrightarrow{a}$ are in Δ'. We say that $\sim$ is a *congruence for* Δ' (or that transitions in Δ' *preserve* $\sim$) if for all $q \sim q'$ and $a \in \Sigma$, if $q \xrightarrow{a} p \in \Delta'$ then there exists $q' \xrightarrow{a} p' \in \Delta'$, and for all such transitions $p \sim p'$. If $\Delta' = \Delta$, we just say that $\sim$ is a *congruence*. We say that $\sim$ is a *strong congruence for* Δ' if, moreover, we have the equality $p = p'$ for transitions as above.

REMARK 2.11. If $\mathcal{A}$ is deterministic and $\sim$ is a congruence for Δ', then these transitions are uniform over $\sim$-classes.

Let $\sqsubseteq$ be a preorder over Q. We say that transitions in Δ' *are monotone for* $\sqsubseteq$ if for all $q \sqsubseteq q'$ and $a \in \Sigma$, if $q \xrightarrow{a} p \in \Delta'$ then there exists $q' \xrightarrow{a} p' \in \Delta'$ and for all such transitions, $p \sqsubseteq p'$. Transitions in Δ' are said strictly monotone for $\sqsubseteq$ if, moreover, whenever $q < q'$, $q \xrightarrow{a} p \in \Delta'$ and $q' \xrightarrow{a} p' \in \Delta'$, we have $p < p'$. If $\Delta' = \Delta$, we simply say that $(\mathcal{A}, \sqsubseteq)$ is *(strictly) monotone*.

All these properties can equivalently be stated with words $w \in \Sigma^*$ instead of letters $a \in \Sigma$.

REMARK 2.12. If transitions in Δ' are monotone for a preorder $\sqsubseteq$, then its induced equivalence relation is a congruence for Δ'.

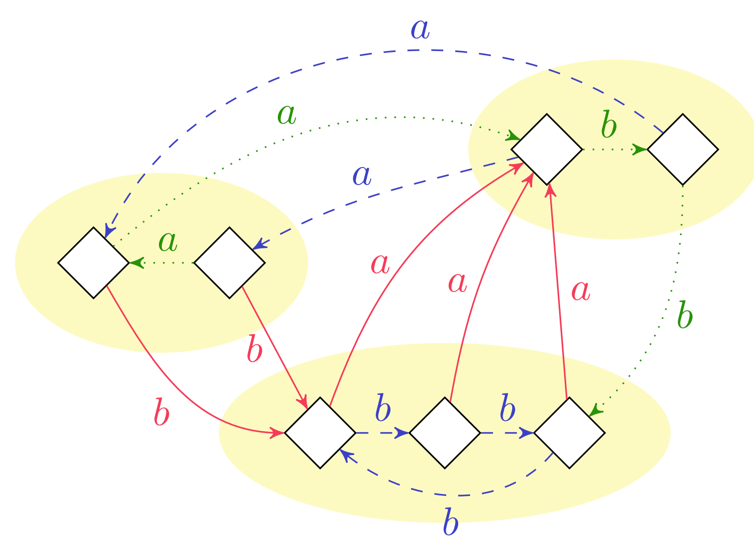

Figure 3. Representation of the notions of uniformity, congruence and strong congruence. We picture an automaton with three equivalence classes, each of them represented by a yellow bubble. Green-dotted transitions are uniform over the classes, but the relation is not a congruence for them. The relation is a congruence for blue-dashed transitions, and a strong congruence for red-solid transitions.

Quotient by a congruence. Let $\mathcal{A}$ be an automaton and let $\sim$ be a congruence over its set of states Q. We define the *quotient of $\mathcal{A}$ by $\sim$* to be the automaton structure $\mathcal{A}/_\sim$ given by:

- The set of states are the $\sim$-classes.
- There is a transition $[q] \xrightarrow{a} [p]$ if there are $q' \in [q]$, $p' \in [p]$ such that $q' \xrightarrow{a} p'$ in $\mathcal{A}$.
- The initial state is $[q_{\mathsf{init}}]$.

We note that if $\sim$ comes from a monotone preorder, the obtained automaton structure $\mathcal{A}/_\sim$ with the induced order over the classes is monotone.

REMARK 2.13. The quotient $\mathcal{A}/_\sim$ is a deterministic automaton structure.

A run over a word w in $\mathcal{A}$, $\rho = q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \dots$ naturally induces a run over w in $\mathcal{A}/_\sim$, $[q_0] \xrightarrow{w_0} [q_1] \xrightarrow{w_1} \dots$, that we call the *projection of ρ* in the quotient automaton.

LEMMA 2.14. *Let $\sim$ be a congruence in $\mathcal{A}$. Any run in $\mathcal{A}/_\sim$ is the projection of some run in $\mathcal{A}$.*

PROOF. Let $[q_0] \xrightarrow{w_0} [q_1] \xrightarrow{w_1} \dots$ be a run in $\mathcal{A}/_\sim$. We build the desired run in $\mathcal{A}$ recursively. For the base case, it suffices to take $p_0 \in [q_0]$ to be the initial state of $\mathcal{A}$ (which belongs to $[q_0]$ by definition of the initial state of $\mathcal{A}/_\sim$). Suppose that $p_0 \xrightarrow{w_0} p_1 \xrightarrow{w_1} \dots p_k$ has already been built, with $p_i \in [q_i]$. By definition of the quotient automaton, there are $q'_k \in [q_k]$ and $q'_{k+1} \in [q_{k+1}]$ with $q'_k \xrightarrow{w_k} q'_{k+1}$. By the definition of a congruence, there is a transition $p_k \xrightarrow{w_k} p_{k+1}$ and $p_{k+1} \in [q_{k+1}]$. ∎

Notations on paths in automata with a congruence. Let $\sim$ be a congruence over the states of a parity automaton $\mathcal{A}$. We write $[q] \xrightarrow{a:x} [p]$ if for all $q' \sim q$, every a-transition from q' is of the form $q' \xrightarrow{a:x} p'$ with $p' \sim p$. We extend this notation to paths $[q] \overset{w:x}{\rightsquigarrow} [p]$ and for outputs $\leq x$, $<x$, $\geq x$ and $>x$ in the natural way.

REMARK 2.15. If $\sim$ is a congruence for transitions producing priority x, $q \overset{w:x}{\rightsquigarrow} p$ implies $[q] \overset{w:x}{\rightsquigarrow} [p]$.

Let $\mathcal{A}$ be a history-deterministic automaton with initial state q_{init} and let r be a resolver for it. We recall that we write $q_{\mathsf{init}} \overset{u:x}{\rightsquigarrow}_{\mathsf{r}} q$ to denote the run induced by r over u. We use the same conventions as above regarding outputs with the symbols $\leq x$, $<x$, $\geq x$ and $>x$.

For $u_0 \in \Sigma^*$, we write $q \overset{w:x}{\rightsquigarrow}_{u_0,\mathsf{r}} p$ if:

- $q_{\mathsf{init}} \overset{u_0}{\rightsquigarrow}_{\mathsf{r}} q$, and
- the induced run of r over $u_0 w$ ends in p and produces x as minimal priority in the part of the run corresponding to w.

We write $q \overset{w:x}{\rightsquigarrow}_{\exists,\mathsf{r}} p$ if $q \overset{w:x}{\rightsquigarrow}_{u_0,\mathsf{r}} p$ for some $u_0 \in \Sigma^*$. We write $q \overset{w:x}{\rightsquigarrow}_{\forall,\mathsf{r}} p$ if, for any word $u_0 \in \Sigma^*$ such that $q_{\mathsf{init}} \overset{u_0}{\rightsquigarrow}_{\mathsf{r}} q$, we have $q \overset{w:x}{\rightsquigarrow}_{u_0,\mathsf{r}} p$.

If $\sim$ is a congruence in $\mathcal{A}$, we write $[q] \overset{w:x}{\rightsquigarrow}_{\forall,\mathsf{r}} [p]$ if, for any word $u_0 \in \Sigma^*$ such that $q_{\mathsf{init}} \overset{u_0}{\rightsquigarrow}_{\mathsf{r}} q' \in [q]$, we have $q' \overset{w:x}{\rightsquigarrow}_{u_0,\mathsf{r}} p' \in [p]$. We avoid using this notation for paths quantified existentially, as we consider that the corresponding semantics are not as intuitive.

2.2.3 Residuals and semantic determinism

Residuals of a language. Let $L \subseteq \Sigma^\omega$ be a language of infinite words and let $u \in \Sigma^*$. We define the *residual of L with respect to u* by

$$u^{-1}L = \{w \in \Sigma^\omega \mid uw \in L\}.$$

We denote $\mathsf{Res}(L)$ the set of residuals of L, which we will always order by inclusion. This induces an equivalence relation $\sim_L$ over Σ^* given by the equality of residuals. The corresponding equivalence classes $[u] = \{u' \in \Sigma^* \mid u^{-1}L = u'^{-1}L\}$ are called *residual classes*. We write $[u] \leq [u']$ if $u^{-1}L \subseteq u'^{-1}L$ (and $<$ if this inclusion is strict).

REMARK 2.16. If L is ω-regular, $\mathsf{Res}(L)$ is finite, and for all $u \in \Sigma^*$, $u^{-1}L$ is also ω-regular. Contrary to the case of finite words, there are non ω-regular languages with a finite set of residuals.

We now state a key monotonicity property for residuals; its proof is a direct check.

LEMMA 2.17. *For any language $L \subseteq \Sigma^\omega$ and for any finite words $u, u', w \in \Sigma^*$, if $[u] \leq [u']$ then $[uw] \leq [u'w]$. In particular, if $[u] = [u']$ then $[uw] = [u'w]$.*

Prefix-independence. We say that a language $L \subseteq \Sigma^\omega$ is *prefix-independent*[4] if for all $w \in \Sigma^\omega$ and $u \in \Sigma^*$, $uw \in L$ if and only if $w \in L$. Equivalently, L is prefix-independent if and only if $\mathsf{Res}(L)$ is a singleton.

Semantic determinism. We say that an automaton $\mathcal{A}$ is *semantically deterministic* if for all state $q \in Q$, letter $a \in \Sigma$ and transitions $q \xrightarrow{a} p_1$ and $q \xrightarrow{a} p_2$, it is satisfied that $\mathcal{L}(\mathcal{A}_{p_1}) = \mathcal{L}(\mathcal{A}_{p_2})$, where $\mathcal{A}_p$ is the automaton obtained by setting p as initial state.

LEMMA 2.18 ([41]). *Any history-deterministic automaton contains an equivalent semantically deterministic and history-deterministic subautomaton. Moreover, for parity automata, this subautomaton can be computed in polynomial time.*

GLOBAL HYPOTHESIS. We assume in the whole paper that history-deterministic automata are semantically deterministic.

We refer to [50] for more details on semantically deterministic automata.

Residual associated to a state. We write $q \sim_{\mathcal{A}} q'$ if $\mathcal{L}(\mathcal{A}_q) = \mathcal{L}(\mathcal{A}_{q'})$ (and drop the subscript if $\mathcal{A}$ is clear from the context). The *class of* q, written $[q]$, is the set of states equivalent to q.

If $\mathcal{A}$ is semantically deterministic and q is reachable, $\mathcal{L}(\mathcal{A}_q)$ coincides with $u^{-1}L$ for any word $u \in \Sigma^*$ leading to q from the initial state of $\mathcal{A}$. In that case, we say that the class of states $[q]$ is *associated to* the residual class $[u]$. The inclusion of these languages induces a preorder $\sqsubseteq_{\mathcal{A}}$ on the states of $\mathcal{A}$ (with its corresponding equivalence relation). By Lemma 2.17, if $\mathcal{A}$ is semantically deterministic, the relation $\sim_{\mathcal{A}}$ is a congruence and the preorder $\sqsubseteq_{\mathcal{A}}$ makes $\mathcal{A}$ a monotone automaton.

REMARK 2.19. An automaton $\mathcal{A}$ without unreachable states is semantically deterministic if and only if $\sim_{\mathcal{A}}$ is a congruence.

Automaton of residuals. Let $L \subseteq \Sigma^\omega$ be a language of infinite words. The *automaton of residuals* of L is a deterministic automaton structure $\mathcal{R}_L$ over Σ defined as follows:

- The set of states is the set of residual classes of $\mathsf{Res}(L)$: $Q = \{[u] \mid u \in \Sigma^*\}$.
- The initial state is $[\varepsilon]$.
- For each state $[u]$ and letter $a \in \Sigma$, it contains the transition $[u] \xrightarrow{a} [ua]$.

We will be interested in the question of whether we can define a parity or Büchi automaton on top of the $\mathcal{R}_L$ so that the obtained automaton recognises L.

The states of $\mathcal{R}_L$ are ordered by the inclusion of residuals. By Lemma 2.17, transitions of $\mathcal{R}_L$ are monotone for this order.

4 In some parts of the litterature, prefix-independence is referred to as shift-invariance.

REMARK 2.20. For any semantically deterministic automaton $\mathcal{A}$ recognising L, the automaton of residuals $\mathcal{R}_L$ coincides with the quotient of $\mathcal{A}$ by the congruence $\sim_{\mathcal{A}}$.

3. Positionality of ω-regular objectives: Statement of the results

In this section, we state the central result of the paper and its consequences: a characterisation of deterministic parity automata recognising positional ω-regular languages (Theorem 3.1). The statement of the theorem uses terminology that will be formally introduced in Section 5, here we just provide some intuitive explanations.

3.1 Characterisation of positionality for ω-regular objectives

We state our main characterisation theorem. Items are ordered following the sequence of logical implications in its proof (with the exception of (3”)).

THEOREM 3.1. *Let $W \subseteq \Sigma^\omega$ be an ω-regular objective. The following are equivalent:*

1. *W is positional over finite ε-free Eve-games.*
2. *There is a deterministic fully progress consistent signature automaton recognising W.*
3. *There is a deterministic ε-completable parity automaton recognising W.*

3’. *There is a history-deterministic ε-complete parity automaton recognising W.*

3”. *Any (non-deterministic) parity automaton recognising W is ε-completable.*

4. *For all cardinals κ, there is a well-ordered monotone (κ, W)-universal graph.*
5. *W is positional over all games (potentially infinite and containing ε-edges).*

This is an automata-oriented characterisation of positionality: we identify two classes of deterministic parity automata (fully progress consistent signature and ε-completable), such that any positional language can be recognised by automata in these classes. Each of them presents some formal advantages that make them suitable for different kinds of proofs. Signature automata are parity automata with a very restricted syntactic structure: for all priorities x there is a total preorder $\sqsubseteq_x$ over the states, such that these refine one another and satisfy some local monotonicity properties (see Section 5.1 for the precise definition, which is quite involved). Our main technical contribution is to show that any positional ω-regular objective W can be recognised by a signature automaton (implication from (1) to (2)). This is achieved by applying a number of transformations to a given parity automaton, until obtaining an automaton with all the desired structural properties. The final automaton satisfies a further more global property – necessary for positionality – that we call full progress consistency: words making a strict progress in the automaton with respect to some of the preorders must be accepted if repeated infinitely often.

Next, we prove that deterministic fully progress consistent signature automata are in fact ε-completable: one may add ε-transitions along a tree structure without augmenting their

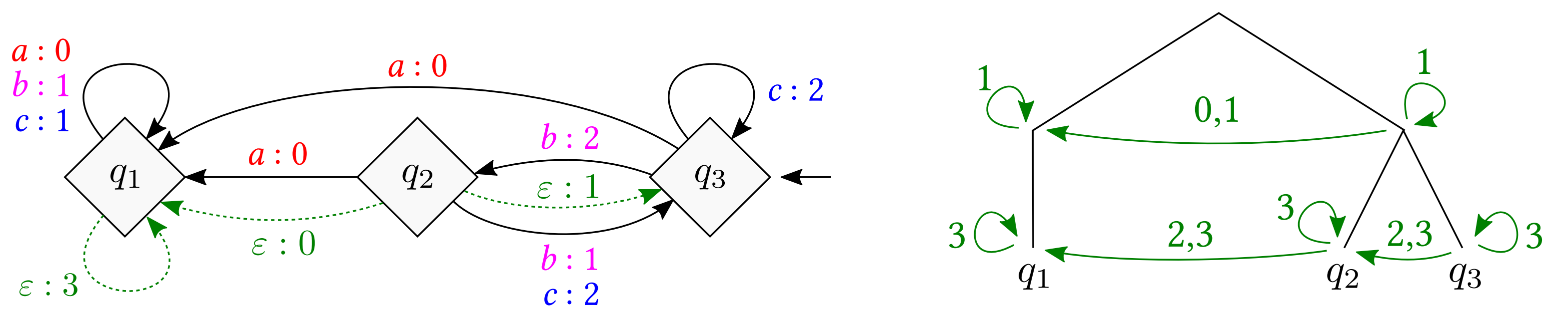

Figure 4. On the left, a deterministic automaton recognising the positional language $\mathtt{Inf}(a) \vee (\mathtt{No}(a) \wedge \mathtt{Fin}(bb))$. On the right, a representation of an ε-completion of the automaton: we can add ε-transitions indicated by the tree, that is: $q_2, q_3 \xrightarrow{\varepsilon:0,1} q_1$, $q_2 \xrightarrow{\varepsilon:1} q_3$, $q_3 \xrightarrow{\varepsilon:1} q_2$, $q_3 \xrightarrow{\varepsilon:2,3} q_2 \xrightarrow{\varepsilon:2,3} q_1$, and $q \xrightarrow{\varepsilon:x} q$ for all q and odd x. Some of these are represented on the left as dotted arrows.

language, as illustrated in Figure 4 (see Section 5.3 for a formal definition). This corresponds to the implication from (2) to (3), Item (3') follows immediatly. We then show how to obtain well-ordered monotone universal graphs from history-deterministic ε-complete automata (implication from (3') to (4)), which is fairly straightforward, and obtain the implication (4) $\Rightarrow$ (5) thanks to Proposition 2.2.

This shows the equivalence of all the statements of Theorem 3.1, except for Item (3"): *any* parity automaton recognising a positional language is ε-completable (including non-deterministic ones). The proof of this result, included in Corollary 6.5, relies on the equivalence between Items (1) and (5) and its consequences (mainly Theorem 3.4 about closure under union). We do not know whether the existence of a non-deterministic ε-complete automaton suffices to prove positionality.

3.2 Main consequences on positionality

We now discuss consequences of Theorem 3.1.

Decidability of positionality in polynomial time

THEOREM 3.2. *Given a deterministic parity automaton $\mathcal{A}$, we can decide in polynomial time whether $\mathcal{L}(\mathcal{A})$ is positional.*

We will give two proofs for Theorem 3.2, both of which are detailed in Section 6. The first proof applies the procedure turning a deterministic parity automaton into a signature automaton; if some step of the procedure fails, then the objective is not positional. The second proof is more direct and builds up on another consequence of Theorem 3.1, namely the closure under union (see discussion below).

Finite-to-infinite and 1-to-2 player lifts The following result simply restates the implication (1) $\implies$ (5) from Theorem 3.1.

THEOREM 3.3. *If an ω-regular objective is positional over finite, ε-free Eve-games, then it is positional over all games (potentially infinite and containing ε-edges).*

Closure under union of prefix-independent positional languages We now show that Kopczyński's conjecture holds for ω-regular languages: prefix-independent positional languages are closed under union. In fact, we show a stronger result: it suffices to suppose that only one of the objectives is prefix-independent.

THEOREM 3.4. *Let $W_1, W_2 \subseteq \Sigma^\omega$ be two positional ω-regular objectives, and suppose that W_1 is prefix-independent. Then, $W_1 \cup W_2$ is positional.*

In order to obtain this theorem, we use the 1-to-2-players lift stated in Theorem 3.3. The result from Theorem 3.4 can be easily obtained for Eve-games (one only needs to be careful with the definition of a uniform strategy), so it suffices then to apply the lift to get the result for all types of games.

LEMMA 3.5. *Let $W_1, W_2 \subseteq \Sigma^\omega$ be two objectives that are positional over Eve-games, and suppose that W_1 is prefix-independent. Then, $W_1 \cup W_2$ is positional over Eve-games.*

PROOF. Let $\mathcal{G}$ be an Eve-game using $W_1 \cup W_2$ as winning condition. We show that Eve has a positional strategy that wins from any vertex of her winning region. We let $\mathcal{G}_1$ be the game with the same game graph than $\mathcal{G}$ and W_1 as winning condition. Consider Eve's winning region in this game, $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$. By positionality of W_1, she has a positional strategy strat_1 ensuring to produce paths labelled with W_1 from states in $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$. Moreover, by prefix-independence of W_1, there is no path leading to $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$ from a vertex that is not in this winning region.

We let $\mathcal{G}_2$ be the game with $\mathcal{G} \setminus \mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$ as game graph, and using W_2 as winning condition. By positionality of W_2, Eve has a positional strategy strat_2 for this game that is winning from $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_2)$.

We consider the positional strategy strat in $\mathcal{G}$ that coincides with strat_1 over $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$ and coincides with strat_2 over $\mathcal{G}_2$. It is clear that this strategy is winning from vertices in $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1) \cup \mathcal{W}in_{\text{Eve}}(\mathcal{G}_2)$. We show that these are all vertices from which Eve can win $\mathcal{G}$, so strat is an optimal strategy.

CLAIM 3.6. $\mathcal{W}in_{\text{Eve}}(\mathcal{G}) = \mathcal{W}in_{\text{Eve}}(\mathcal{G}_1) \cup \mathcal{W}in_{\text{Eve}}(\mathcal{G}_2)$.

Proof. Let v be a vertex in $\mathcal{G}$ such that Eve wins from it. A strategy in an Eve-game is just an infinite path in the game graph. Let therefore ρ_v be an infinite path from v labelled with a word $w \in W_1 \cup W_2$. Suppose that $v \notin \mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$. In particular $w \notin W_1$, so $w \in W_2$. As there is no path leading to $\mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$ from a vertex that is not in this region, the path ρ_v is contained in $\mathcal{G} \setminus \mathcal{W}in_{\text{Eve}}(\mathcal{G}_1)$, which is the game graph of $\mathcal{G}_2$. Therefore, $v \in \mathcal{W}in_{\text{Eve}}(\mathcal{G}_2)$. ◆

This finishes the proof. ■

Kopczyński's conjecture and its stronger version in which only one of the objectives is supposed to by prefix-independent remain open for arbitrary (non ω-regular) objectives.

Closure of positionality under addition of neutral letters As mentioned in the introduction, Ohlmann recently characterised positional objectives by means of the existence of universal graphs [47]. One direction (stated in Proposition 2.2) holds for any objective: if W admits well-ordered monotone universal graphs, then it is positional. To obtain the converse, the proof proposed by Ohlmann requires a further hypothesis: W has to contain a neutral letter, that is, a letter that can be removed from any word without modifying the membership in W. In his work, he left open the problem of whether adding a neutral letter preserves positionality. This is a central question in the theory of positionality, as it would imply that universal graphs completely characterise positionality without any further hypothesis on the objectives. This question is almost[5] equivalent to the one raised by Kopczyński in his PhD thesis [34, Section 2.5]: if W is positional over ε-free games, is it positional over all games? We introduce these notions formally for completeness.

Let $W \subseteq \Sigma^\omega$ be an objective. A letter $c \in \Sigma$ is *neutral for W* if, for all $w_1, w_2, \dots \in \Sigma^+$ and $n_1, n_2, \dots \in \mathbb{N}$:

— $c^{n_1} w_1 c^{n_2} w_2 \cdots \in W \iff w_1 w_2 \cdots \in W$, and
— $w_1 c^\omega \in W \iff {w_1}^{-1} W \neq \emptyset$.

Given an objective W, we let W^ε denote the unique objective obtained by adding a fresh neutral letter ε to W.

PROPOSITION 3.7 ([47]). *Let $W \subseteq \Sigma^\omega$. The objective W^ε is positional if and only if for all cardinals κ there is a well-ordered monotone (κ, W)-universal graph.*

CONJECTURE 3.8 (*Neutral letter conjecture* [47]). *For every positional objective W, the objective W^ε is positional.*

Our characterisation (Item (4) in Theorem 3.1), together with Proposition 3.7, answers this question in the case of ω-regular objectives.

THEOREM 3.9. *Let $W \subseteq \Sigma^\omega$ be an ω-regular objective. If W is positional, then W^ε is positional. Also, W is positional over ε-free games if and only if W is positional over all games.*

5 The only difference is that in ε-free games we assume that there are no infinite paths composed exclusively of ε-edges, whereas these may appear in W^ε-games.

4. Warm-up: Illustrating ideas on restricted classes of languages

The goal of this section is to give a gentle introduction to the techniques and ideas which are used in the proof of our main result (implication from (1) to (2) in Theorem 3.1). Readers who prefer to go directly to the statement and proofs of the general case may skip this section.

We single out four crucial properties that a parity automaton recognising a positional objective should satisfy; the general characterisation will consist in a generalisation of these.

— Positional objectives have residuals totally ordered by inclusion (inducing a total order on the corresponding residual classes of the states of automata).
— This order should satisfy a semantic property called progress consistency.
— Transitions with priority 0 preserve the congruence induced by the residuals (if $q \sim p$ and $q \xrightarrow{a:0}$, then $p \xrightarrow{a:0}$).
— In each congruence class, states which are interreachable using paths avoiding priorities ≤ 1 have comparable (≤ 1)-safe languages (defined below).

We propose to study four restricted classes of ω-regular objectives that allow us to isolate these different points, namely, closed, open, Büchi recognisable and coBüchi recognisable objectives. Considering objectives in these four classes allows us to illustrate the necessity of the four properties above, and the techniques we use to derive them. In each case, we state a characterisation of positionality and give a full proof of necessity, which is the more difficult direction. These characterisation and proof techniques are generalised to all ω-regular languages in our main inductive proof of necessity (Section 5.2).

We moreover incorporate in this section many examples illustrating our results and the ideas in our proofs.

4.1 Closed objectives and total order on the residuals

We now discuss the first property announced above: residuals of positional objectives are totally ordered by inclusion. The necessity of this condition holds even for non ω-regular objectives.

Residuals of positional objectives are totally ordered

LEMMA 4.1. *If an objective $W \subseteq \Sigma^\omega$ is positional, then* $\mathsf{Res}(W)$ *is totally ordered by inclusion.*

PROOF. We show the contrapositive. Suppose that W has two incomparable residuals, $u_1{}^{-1}W$ and $u_2{}^{-1}W$. Take $w_1 \in u_1^{-1}W \setminus u_2^{-1}W$ and $w_2 \in u_2^{-1}W \setminus u_1^{-1}W$. Stated differently, we have

$$u_1 w_1 \in W, \quad u_1 w_2 \notin W,$$
$$u_2 w_1 \notin W, \quad u_2 w_2 \in W.$$

Consider the (infinite) Eve-game $\mathcal{G}$ represented in Figure 5. Eve wins $\mathcal{G}$ from v_1 and v_2: if a play starts in v_i, for $i = 1, 2$, she just has to take the path labelled w_i from v_{choice}. However, she

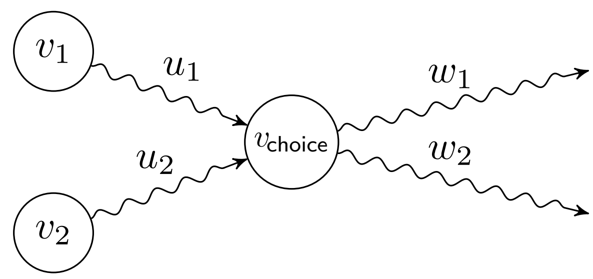

Figure 5. A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if $\text{Res}(W)$ is not totally ordered, as in the proof of Lemma 4.1.

cannot win from both v_1 and v_2 using a positional strategy. Indeed, such a positional strategy would choose one transition $v_{\text{choice}} \overset{w_i}{\rightsquigarrow}$, and the play induced when starting from v_j, $j \neq i$, would be losing. ∎

Closed objectives Let Σ be a set of letters and $L \subseteq \Sigma^*$ be a language of finite words. The safety objective associated to L is defined by

$$\text{Safety}(L) = \{w \in \Sigma^\omega \mid w \text{ does not contain any prefix in } L\}.$$

An objective W is *topologically closed* if $W = \text{Safety}(L)$ for some $L \subseteq \Sigma^*$. This terminology is justified since objectives of the form $\text{Safety}(L)$ are exactly the closed subsets of Σ^ω for the Cantor topology (see for example [53]).

REMARK 4.2. An objective $W = \text{Safety}(L)$ is ω-regular if and only if L is a regular language of finite words if and only if $\text{Res}(W)$ is finite. We refer to this class as *ω-regular closed objectives.*

It turns out that for ω-regular closed objectives the converse of Lemma 4.1 holds. This was first established in [25].

PROPOSITION 4.3 (Positionality of closed objectives [25]). *Let $W \subseteq \Sigma^\omega$ be an ω-regular closed objective. Then, W is positional if and only if* $\text{Res}(W)$ *is totally ordered by inclusion.*

Thus, residuals encode the information needed to decide whether an ω-regular closed objective is positional. We do not include a proof of sufficiency in this warm-up; a proof for all (non-necessarily ω-regular) closed objectives is given in Theorem 8.4. However, a much subtler understanding is needed for non-closed objectives, as witnessed by the example below.

EXAMPLE 4.4 (Non-positional open objective). Consider the non-closed objective

$$W = \{w \in \Sigma^\omega \mid w \text{ contains the factor } aa\}.$$

Its three residuals are totally ordered by inclusion:

$$\varepsilon^{-1}W \subseteq a^{-1}W \subseteq (aa)^{-1}W.$$

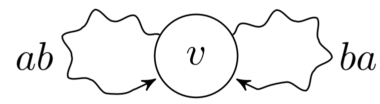

Figure 6. A game $\mathcal{G}$ in which Eve cannot produce the factor aa positionally.

However, it is not positional, as witnessed by the game in Figure 6. ◆

4.2 Open objectives and progress consistency

We now introduce progress consistency, a semantic property of the order of residuals which is necessary for positionality. For ω-regular open objectives, this property, together with the total order of residuals is also sufficient.

Progress consistency

DEFINITION 4.5 (Progress consistency). An objective $W \subseteq \Sigma^\omega$ is *progress consistent* if for all $u, w \in \Sigma^*$:

$$[u] < [uw] \implies uw^\omega \in W.$$

Intuitively, a progress consistent objective satisfies that whenever we read a word that makes some strict progress with respect to the order of the residuals, by repeating this word we produce a sequence in W.

We remark that the objective $\mathrm{Reach}(\Sigma^* aa)$ from Example 4.4 is not progress consistent, as the word ba makes progress from residual $\varepsilon^{-1}W$, but $(ba)^\omega \notin W$.

Let us establish necessity of progress consistency for positional objectives.

LEMMA 4.6 (Necessity of progress consistency). *Any positional objective is progress consistent.*

PROOF. We show the contrapositive of the statement. Let W be an objective that is not progress consistent, that is, there are $u, w \in \Sigma^*$ such that $[u] < [uw]$ and $uw^\omega \notin W$. Let $w' \in (uw)^{-1}W \setminus u^{-1}W$. Consider the game $\mathcal{G}$ depicted in Figure 7.

Eve wins game $\mathcal{G}$ from vertex v_0 by producing the play

$$v_0 \overset{u}{\rightsquigarrow} v_{\text{choice}} \overset{w}{\rightsquigarrow} v_{\text{choice}} \overset{w'}{\rightsquigarrow}.$$

However, she cannot win positionally from v_0 since positional strategies produce either uw^ω or uw', and both of these words are losing. ■

REMARK 4.7. The previous lemma applies, in particular, to ω-regular closed objectives. We did not need to add progress consistency as a hypothesis in Proposition 4.3, as this property is granted for closed objective by Lemma 2.17.

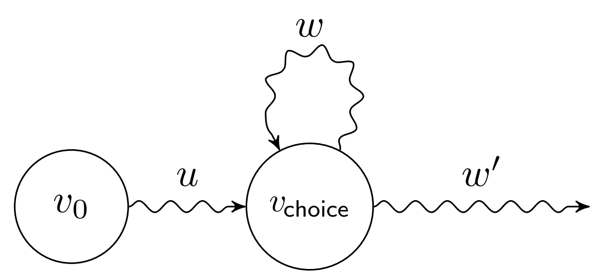

Figure 7. A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if W is not progress consistent, as in the proof of Lemma 4.6.

We are now ready to move on to the characterisation of ω-regular open objectives.

Open objectives We now study the dual of closed objectives, namely, open ones. Let $L \subseteq \Sigma^*$. The reachability objective associated to L is defined by

$$\mathsf{Reach}(L) = \{w \in \Sigma^\omega \mid w \text{ contains a prefix in } L\}.$$

An objective W is *topologically open* if $W = \mathsf{Reach}(L)$ for some $L \subseteq \Sigma^*$. (These are the open subsets of Σ^ω for the Cantor topology.) Similarly to the previous subsection, we define the class of *ω-regular open objectives* as those that are both open and ω-regular.

REMARK 4.8. An open objective $W = \mathsf{Reach}(L)$ is ω-regular if and only if L is a regular language of finite words if and only if $\mathsf{Res}(W)$ is finite.

Let us state a characterisation of positionality for ω-regular open objectives. Characterisations for the full classes of open and closed objectives (without ω-regularity assumptions) will be obtained in Section 8.

PROPOSITION 4.9 (Positionality for open objectives). *An ω-regular open objective W is positional if and only if it is progress consistent and its set of residuals* $\mathsf{Res}(W)$ *is totally ordered.*

Necessity follows from combining Lemmas 4.1 and 4.6; we omit a proof of sufficiency in this warm-up. In particular, we obtain the following corollary of Propositions 4.3 and 4.9.

COROLLARY 4.10. *Any positional ω-regular open objective is bipositional.*

We now give an example of an objective that satisfies the requirement from the previous proposition.

EXAMPLE 4.11 (Positional open objective). Consider the ω-regular open objective

$$W_n = \mathsf{Reach}((a\Sigma^*)^n).$$

It was introduced (for $n = 2$) in [3, Lemma 13] as an example of a bipositional objective which is not *concave* (that is, no shuffle of two words outside W belongs to W; see [34, Def 4.2] for details). Its residuals are given by

$$\varepsilon^{-1}W \subsetneq a^{-1}W \subsetneq (aa)^{-1}W \subsetneq \cdots \subsetneq (a^n)^{-1}W = \Sigma^\omega,$$

which are totally ordered. Moreover, for any residual class $[a^i]$ with $i < n$, we have $[a^i] < [a^i u]$ if and only if u contains the letter a, in which case $a^i u^\omega \in W$. Therefore, W is progress consistent, so we conclude that it is bipositional. ◆

Many natural examples of objectives are in fact prefix-independent; for those, the two conditions about the residuals above are trivially satisfied. Yet, this does not suffice to guarantee their positionality. We continue our introductory exploration with objectives recognised by deterministic Büchi automata.

4.3 Büchi recognisable objectives: Uniformity of 0-transitions

Our goal in this section is to present another property of positional ω-regular objectives, namely, that they can be recognised by a deterministic parity automaton $\mathcal{A}$ in which 0-transitions are uniform over each residual class:

$$\text{For any } q \sim_{\mathcal{A}} q' \text{ and } a \in \Sigma, \text{ if } q \xrightarrow{a:0} \text{ then } q' \xrightarrow{a:0}.$$

In our main induction (Section 5.2), we will derive a similar property for all even priorities. To illustrate the technique, we now only focus on the case of Büchi recognisable languages, which helps alleviate some of the technicalities while preserving the important ideas behind the proof. On the way, we characterise positionality for these objectives, reobtaining the main result of [8].

The proof is split into two parts: first, we focus on the prefix-independent case, and then reduce to it.

Prefix-independent Büchi recognisable objectives It is well-known that *Büchi languages* are positional [30], that is, those of the form

$$\mathsf{Buchi}_\Sigma(B) = \{w \in \Sigma^\omega \mid \text{ letters of } B \text{ appear infinitely often in } w\}.$$

We prove now that these are the only positional prefix-independent Büchi recognisable objectives. Our proof considerably simplifies that of [8].

PROPOSITION 4.12 ([8, Proposition 11]). *A prefix-independent Büchi recognisable W is positional if and only if it is a Büchi language.*

In particular, Proposition 4.12 tells us that there is a Büchi automaton with just one state recognising W. In this automaton, 0-transitions are trivially uniform.

Super words and super letters. We say that $u \in \Sigma^+$ is a *super word* (for W) if, for every $w \in \Sigma^\omega$, if w contains u infinitely often as a factor, then $w \in W$. If u is a letter, we say that it is a *super letter*. Let $B_W \subseteq \Sigma$ be the set of super letters for W. It is clear that $\mathsf{Buchi}_\Sigma(B_W) \subseteq W$. We will show that, if W is positional, this is in fact an equality.

LEMMA 4.13 (Existence of super letters). *A non-empty prefix-independent positional objective W recognised by a deterministic Büchi automaton admits a super letter.*

One may easily deduce Proposition 4.12 from Lemma 4.13: the restriction W' of W to non-super letters is a prefix-independent Büchi recognisable positional objective which contains no super letter. Thus, Lemma 4.13 tells us that $W' = \emptyset$ and therefore $W = \mathsf{Buchi}_\Sigma(B_W)$.

PROOF OF LEMMA 4.13. Fix a non-empty prefix-independent positional objective W recognised by a deterministic Büchi automaton $\mathcal{A}$, and assume without loss of generality that $\mathcal{A}$ in normal form and strongly connected, which can be assumed by prefix-independence (thus, every state can be chosen initial). Since W is non-empty, note that $\mathcal{A}$ must contain a transition with priority 0. We will use the following observation.

CLAIM 4.14 (Super words in Büchi automata). *A word $w \in \Sigma^+$ is a super word if and only if for all states q of $\mathcal{A}$, priority 0 appears on the path $q \xrightarrow{w:0}$. This is in particular the case if w is a letter.*

Proof. By normality of $\mathcal{A}$, if there is q such that $q \xrightarrow{w:1} q'$ then there is a word $w' \in \Sigma^*$ labelling a returning path $q' \xrightarrow{w':1} q$. Therefore, $(ww')^\omega \notin W$, so w is not a super word. The converse implication is clear, since each time word w is read, the automaton produces priority 0. ◆

We now prove existence of super words.

CLAIM 4.15 (Existence of super words). *There is a super word for $\mathcal{L}(\mathcal{A})$.*

Proof. We note that, as $\mathcal{A}$ is strongly connected and contains some priority 0, for each state q there is a finite word that produces priority 0 when read from q. We let $\{q_1, q_2, \ldots, q_k\}$ be an enumeration of the states of $\mathcal{A}$ and recursively define k finite words $w_1, w_2, \ldots, w_k \in \Sigma^*$ satisfying:

$$q_i \xrightarrow{w_1 w_2 \ldots w_{i-1}} q' \xrightarrow{w_i:0} q''.$$

In words, for $i \in 1, \ldots, k$, reading $w_1 w_2 \ldots w_i$ from q_i, produces priority 0. This implies that $w_1 w_2 \ldots w_i$ produces priority 0 when read from any q_j for $j \leq i$. Therefore, $w = w_1 w_2 \ldots w_k$ produces priority 0 when read from any state of $\mathcal{A}$, so by Claim 4.14, w is a super word. ◆

We now prove that, by positionality of W, super words can be chopped into smaller super words. This implies Lemma 4.13 by repeatedly chopping a super word obtained from Claim 4.15 until obtaining a super letter.

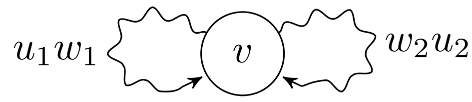

Figure 8. A game $\mathcal{G}$ in which Eve can win by forming the super word $w_1 w_2$ infinitely often, but in which she cannot win using a positional strategy.

CLAIM 4.16 (Chopping super words). *Let $w = w_1 w_2 \in \Sigma^+$ be a super word. Then either w_1 or w_2 is a super word.*

Proof. Suppose by contradiction that neither w_1 nor w_2 are super words. Then, by Claim 4.14, there are states q_1 and q_2 such that $q_1 \xrightarrow{w_1:1} q_1'$ and $q_2 \xrightarrow{w_2:1} q_2'$. By normality, we obtain returning paths $q_1' \xrightarrow{u_1:1} q_1$ and $q_2' \xrightarrow{u_2:1} q_2$. Therefore, $(w_1 u_1)^\omega \notin W$ and $(w_2 u_2)^\omega \notin W$. We consider the game $\mathcal{G}$ depicted in Figure 8. Eve can win this game, as alternating the two self loops she produces the word $(u_1 w_1 w_2 u_2)^\omega$, which belongs to $\mathcal{L}(\mathcal{A})$ since $w_1 w_2$ is a super word. However, positional strategies in this game produce either $(w_1 u_1)^\omega$ or $(w_2 u_2)^\omega$, both losing. This contradicts the positionality of $\mathcal{L}(\mathcal{A})$. ◆

■

Positionality for Büchi recognisable objectives We now state a characterisation of positionality for all Büchi recognisable objectives, without assuming prefix-independence.

PROPOSITION 4.17 (Positionality of Büchi recognisable objectives [8, Theorem 10]). *Let $W \subseteq \Sigma^\omega$ be a Büchi recognisable objective. Then, W is positional if and only if:*

— $\mathsf{Res}(W)$ *is totally ordered,*
— W *is progress consistent, and*
— W *can be recognised by a Büchi automaton on top of the automaton of residuals.*

EXAMPLE 4.18 (Positional Büchi recognisable objective [8, Example 7]). Over the alphabet $\Sigma = \{a, b\}$, let

$$W = \mathtt{Inf}(a) \cup \mathsf{Reach}(aa),$$

that is, a word $w \in \Sigma^\omega$ belongs to W if either it contains letter a infinitely often, or it contains the factor aa at some point. This objective has three different residuals,

$$[\varepsilon] < [a] < [aa] = \Sigma^\omega.$$

Figure 9 depicts a deterministic Büchi automaton defined on top of the residual automaton of W. It is easy to verify that this objective is progress consistent, so by Proposition 4.17, it is a positional objective. ◆

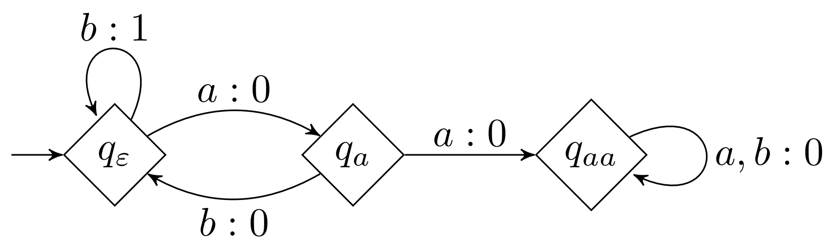

Figure 9. Büchi automaton recognising the objective $W = \mathtt{Inf}(a) \cup \mathsf{Reach}(aa)$.

As earlier, we focus on explaining the necessity of the conditions from Proposition 4.17, and we omit a proof of sufficiency in this warm-up. We already know that the two first conditions are necessary (Lemmas 4.1 and 4.6). We now present the techniques used to obtain the necessity of the third condition.

Uniformity of 0-transitions for positional objectives Our objective is to derive the following result.

LEMMA 4.19 (Uniform behaviour of 0-transitions). *Let $W \subseteq \Sigma^\omega$ be a positional Büchi recognisable language. There is a deterministic Büchi automaton $\mathcal{A}$ recognising W such that for every pair $q \sim_{\mathcal{A}} q'$ of equivalent states and for every letter a, transition $q \xrightarrow{a}$ produces priority 0 if and only if transition $q' \xrightarrow{a}$ produces priority 0.*

Necessity of the conditions from Proposition 4.17 easily follows. In fact, what the previous lemma tells us is that we can take the quotient automaton $\mathcal{A}/_{\sim_{\mathcal{A}}}$ and assign priorities to its transitions consistently.

The techniques we now introduce for proving Lemma 4.19 will be extended in our main induction (Section 5.2). The idea is to reduce to the prefix-independent case, captured by Proposition 4.12. For this, we associate a prefix-independent language, over an ad-hoc alphabet, to each residual of the objective under consideration.

For the rest of this part of the section, we fix a positional objective W recognised by a deterministic Büchi automaton $\mathcal{A}$.

Localising to a residual. For each residual class $[u]$ of W, we define the local alphabet at $[u]$ as:

$$\Sigma_{[u]} = \{w \in \Sigma^+ \mid [uw] = [u] \text{ and for any proper prefix } w' \text{ of } w, [uw'] \neq [u]\}.$$

Note that, if it is non-empty, $\Sigma_{[u]}$ is a prefix code, and therefore it is a uniquely decodable alphabet. Note that in general $\Sigma_{[u]}$ may be infinite, however this is completely harmless in this context, and we will freely allow ourselves to talk about automata over infinite alphabets.[6] Also, $\Sigma_{[u]}$ is possibly empty; in the following definitions we assume that this is not the case.

Seeing words in $\Sigma_{[u]}^\omega$ as words in Σ^ω, define the *localisation of W to* $[u]$ to be the objective

$$W_{[u]} = \{w \in \Sigma_{[u]}^\omega \mid uw \in W\}.$$

Observe that $W_{[u]}$ is prefix-independent. Moreover it is positional: any $W_{[u]}$-game in which Eve could not play optimally using positional strategies would provide a counterexample for the positionality of W.

For a state q of a Büchi automaton $\mathcal{A}$ recognising W, define $\Sigma_{[q]}$ and $W_{[q]}$ in the natural way: $\Sigma_{[q]} = \Sigma_{[u]}$ and $W_{[q]} = W_{[u]}$ for u a word that reaches q from the initial state. Observe that a word $w \in \Sigma^*$ belongs to $\Sigma_{[q]}^*$ if and only if it connects states in the class $[q]$. Elements in $\Sigma_{[q]}$ are those that do not pass twice through this class. We remark that $\Sigma_{[q]} \neq \emptyset$ if and only if q is a recurrent state (it belongs to some non-trivial SCC).

Let q be a recurrent state. The *local automaton of the residual* $[q]$ is the Büchi automaton $\mathcal{A}_{[q]}$ defined as:

- The set of states is $[q]$.
- The initial state is arbitrary.
- For $w \in \Sigma_{[q]}$, $q \xrightarrow{w:x} q'$ if $q \overset{w:x}{\rightsquigarrow} q'$ in $\mathcal{A}$.

The language of $\mathcal{A}_{[q]}$ is $W_{[u]}$, thus $W_{[u]}$ is Büchi recognisable. Therefore, Proposition 4.12 yields that $W_{[u]}$ is a Büchi language: there exists a set $B_{[u]} \subseteq \Sigma_{[u]}$ such that $W_{[u]} = \mathsf{Buchi}_{\Sigma_{[u]}}(B_{[u]})$. We let $N_{[u]} = \Sigma_{[u]} \setminus B_{[u]}$ be the set of non-super letters, and extend these notations to states of $\mathcal{A}$ by putting $B_{[q]} = B_{[u]}$ and $N_{[q]} = N_{[u]}$ where u is any word leading from the initial state to q.

Polished automata. For a recurrent state q, we say that a residual class $[q]$ is *polished in $\mathcal{A}$* if:

1. For all $q_1, q_2 \in [q]$, there is a word $u \in N_{[q]}^*$ such that $q_1 \overset{u:1}{\rightsquigarrow} q_2$.
2. For every $q' \in [q]$ and every word $u \in N_{[q]}$, reading u from q' produces priority 1.

Stated differently, $[q]$ is polished if the restriction of $\mathcal{A}_{[q]}$ to transitions labelled with letters in $N_{[q]}$ is strongly connected and does not contain any transition with priority 0.

We say that the automaton $\mathcal{A}$ is *polished* if all its residual classes are polished.

REMARK 4.20. If $\mathcal{A}$ is polished and q is a transient state, then, it is the only state in its residual class: $[q] = \{q\}$. We will apply the term *recurrent* (resp. *transient*) to a class $[q]$ if q is recurrent (resp. transient). This is well defined by the previous comment.

LEMMA 4.21 (Obtaining a polished automaton). *Any positional Büchi recognisable language W can be recognised by a polished deterministic Büchi automaton.*

6 Note that in a finite automaton $\mathcal{A}$ over an infinite alphabet Σ, there are finitely many classes of letters such that two letters from the same class admit exactly the same transitions in $\mathcal{A}$. We can let Σ_{fin} be the set of equivalence classes of Σ for this relation, and let $\mathcal{A}_{\text{fin}}$ be the induced automaton over Σ_{fin}. It holds, that $w \in \mathcal{L}(\mathcal{A})$ if and only if $w_{\text{fin}} \in \mathcal{L}(\mathcal{A}_{\text{fin}})$ where w_{fin} is the projection of w to Σ_{fin} .

PROOF. Let $\mathcal{A}$ be a deterministic Büchi automaton recognising W. We will first polish the residual class $[q]$ of a fixed state q. Consider the restriction $\mathcal{A}'_{[q]}$ of $\mathcal{A}_{[q]}$ to transitions labelled with $N_{[q]}$, and take $S_{[q]}$ to be a final SCC of $\mathcal{A}'_{[q]}$; without loss of generality we assume that $q \in S_{[q]}$.

Now consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by removing states in $[q] \setminus S_{[q]}$, and *redirecting* transitions that go to $[q] \setminus S_{[q]}$ in $\mathcal{A}$ to transitions towards q producing priority 0. Note that this transformation *preserves the residuals*: if $p_1 \xrightarrow{a} p_2$ in $\mathcal{A}$ and $p_1 \xrightarrow{a} p_2'$ in $\mathcal{A}'$, then $p_2 \sim_{\mathcal{A}} p_2'$. Also, either $|\mathcal{A}'| < |\mathcal{A}|$, or $\mathcal{A}$ is left unchanged. We now prove that it preserves the language.

CLAIM 4.22. *Automaton $\mathcal{A}'$ recognises the objective W.*

Proof. Let $w \in \Sigma^\omega$. Suppose first that the run over w in $\mathcal{A}'$ eventually does not take redirected transitions. Then, this run contains a suffix that is also a run in $\mathcal{A}$. As the transformation preserves the residuals, w is accepted by $\mathcal{A}'$ if and only if it is accepted by $\mathcal{A}'$.

Suppose now that the run over w in $\mathcal{A}'$ takes infinitely many redirected transitions. Such a run in $\mathcal{A}'$ is of the form

$$q_{\mathsf{init}} \overset{w_0}{\rightsquigarrow} q \overset{w_1}{\rightsquigarrow} p_1 \xrightarrow{a_1:0} q \overset{w_2}{\rightsquigarrow} p_2 \xrightarrow{a_2:0} q \overset{w_3}{\rightsquigarrow} p_3 \xrightarrow{a_3:0} \dots,$$

where for all i the transition $p_i \xrightarrow{a_i:0} q$ is a redirected one, meaning that in $\mathcal{A}$, reading a_i from p_i leads to $[q] \setminus S_{[q]}$. Note that w is accepted by $\mathcal{A}'$, so we should prove that $w \in \mathcal{L}(\mathcal{A}) = W$. Observe that, for $i \geq 1$, $w_i a_i \in \Sigma^*_{[q]}$ and reading $w_i a_i$ in $\mathcal{A}$ takes q to $[q] \setminus S_{[q]}$, and thus by definition of $S_{[q]}$, it holds that $w_i a_i \notin N^*_{[q]}$, so $w_i a_i \in N^*_{[q]} B^+_{[q]} N^*_{[q]}$. We conclude that $w_1 a_1 w_2 a_2 \dots \in \mathsf{Buchi}_{\Sigma_{[q]}}(B_{[q]}) = q^{-1}W$ hence $w \in W$. ◆

It follows that the residual class of q in $\mathcal{A}'$ is $[q]_{\mathcal{A}'} = [q]_{\mathcal{A}} \cap Q' = S_{[q]}$. We now prove that it is polished.

CLAIM 4.23. *The residual class $[q]$ is polished in $\mathcal{A}'$.*

Proof. Let $q_1, q_2 \in S_{[q]}$. By definition of $S_{[q]}$ there is $w \in N_{[q]}$ such that $q_1 \overset{w}{\rightsquigarrow} q_2$ in $\mathcal{A}$. As this path avoids $[q] \setminus S_{[q]}$ in $\mathcal{A}$, it also belongs to $\mathcal{A}'$. We show that it produces exclusively priorities 1. By definition of $S_{[q]}$, there is a path $q_2 \overset{w'}{\rightsquigarrow} q_1$ with $w' \in N_{[q]}$. Therefore $(ww')^\omega$ does not belongs to $W_{[q]}$, so the path $q_1 \overset{w}{\rightsquigarrow} q_2$ cannot produce priority 0, which proves the first point in the definition of a polished class.

Now take $q' \in S_{[q]}$ and $u \in N^*_{[q]}$. Then by definition of $S_{[q]}$, reading u from q' in $\mathcal{A}$ leads back to $S_{[q]}$. By the previous argument, the minimal priority in this path is 1. Again, this path avoids $[q] \setminus S_{[q]}$ in $\mathcal{A}$, so it also belongs to $\mathcal{A}'$. ◆

Thus we have obtained an automaton $\mathcal{A}'$ for W in which the class $[q]$ is polished. Since there are finitely many residual classes, and the obtained automaton $\mathcal{A}'$ is strictly smaller than

$\mathcal{A}$, we can repeat the process (normalising the automaton after each iteration, which does not increase the size) until obtaining an automaton in which all the classes are polished. (We remark that we do not claim that classes $[p] \neq [q]$ that were polished in $\mathcal{A}$ will remain polished in $\mathcal{A}'$. Nevertheless, the process reaches a fixpoint in which all classes are polished.) ■

We will later on use the following property, which is our main reason for introducing polished automata.

LEMMA 4.24 (Connection via losing words). *Let $q \sim_{\mathcal{A}} q'$ be two different recurrent equivalent states of a polished automaton $\mathcal{A}$. Then there is a word $u \in \Sigma^+_{[q]}$ such that $q \xrightarrow{u} q'$ and $u^\omega \notin \mathcal{L}(\mathcal{A}_q)$.*

PROOF. As the automaton $\mathcal{A}$ is polished, there is a word $u \in N^+_{[q]}$ such that $q \xrightarrow{u} q'$. This word satisfies the desired requirement. ■

Uniform behaviour of 0-transitions. Proof of Lemma 4.19. Let $\mathcal{A}$ be a polished and normalised deterministic Büchi automaton recognising W, and suppose by contradiction that there are two states $q_1 \sim_{\mathcal{A}} q_2$ and a letter $a \in \Sigma$ such that $q_1 \xrightarrow{a:0} p_1$ and $q_2 \xrightarrow{a:1} p_2$. By normality, there is a word $w \in \Sigma^*$ such that $p_2 \xrightarrow{w:1} q_2$. In particular, since $q_2 \xrightarrow{aw} q_2$, the class $[q_2]$ is recurrent in $\mathcal{A}$. Note that $aw \in \Sigma^*_{[q_1]}$ and $(aw)^\omega \notin q_1^{-1}W$. Let q' be the state such that $p_1 \xrightarrow{w} q'$, note that $q' \in [q_1]$. Let u_0 be such that $q_{\mathsf{init}} \xrightarrow{u_0} q_1$. See Figure 10 for an illustration of the situation.

Since $(aw)^\omega \notin q_1^{-1}W$, and $q_1 \xrightarrow{aw:0} q'$, it cannot be that case that $q' = q_1$. Hence, Lemma 4.24 gives a word $u \in \Sigma^+_{[q]}$ such that $q' \xrightarrow{u} q_1$ and $u^\omega \notin \mathcal{L}(\mathcal{A}_q)$. Together, the facts that

- $(aw)^\omega \notin \mathcal{L}(\mathcal{A}_q)$,
- $u^\omega \notin \mathcal{L}(\mathcal{A}_q)$, and
- $(awu)^\omega \in \mathcal{L}(\mathcal{A}_q)$

prove that Eve wins the game on the right of Figure 10, but not positionally.

Thus we proved that if W is a Büchi recognisable positional objective, it can be recognised by a Büchi automaton in which 0-transitions behave uniformly. By extending the technique from this section, we will show in Section 5.2 that this property (and its generalisation to all even priorities) also holds for automata using higher priorities.

4.4 Objectives recognised by coBüchi automata: Total order given by safe languages

We now consider coBüchi recognisable objectives. Our analysis is based on history-deterministic automata and techniques from [1].

Most of the section is devoted to the case of prefix-independent objectives, for which we propose a characterisation of positionality (Proposition 4.28). At the end of the section, we comment on how to extend this characterisation to any coBüchi recognisable objective (Proposition 4.38). In the spirit of this warm-up, we omit proofs of sufficiency.

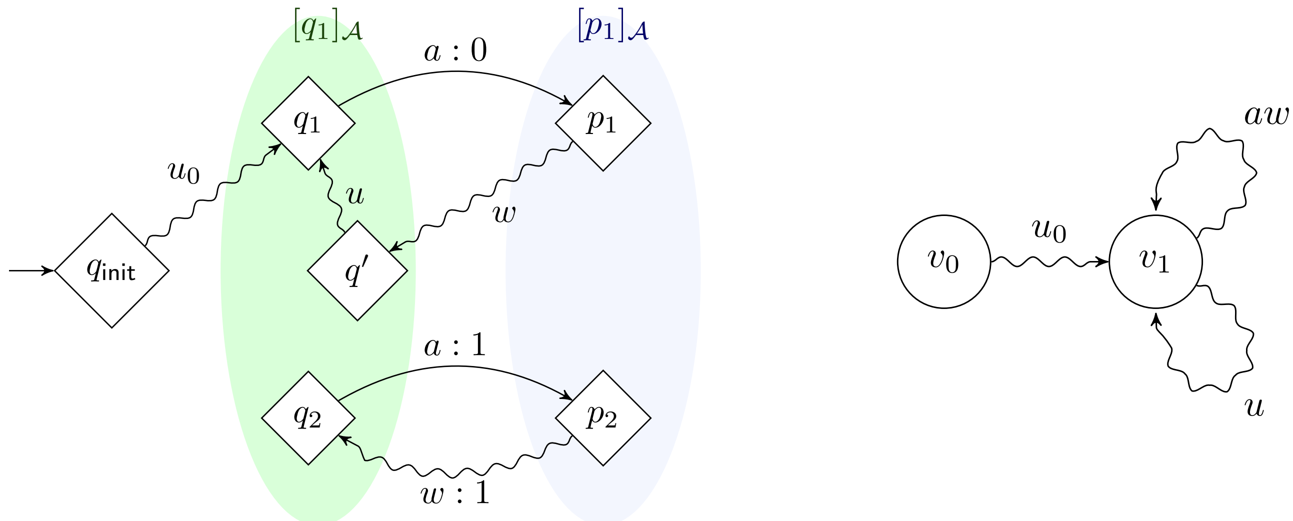

Figure 10. On the left, the situation in the proof of Lemma 4.19. We have the equivalences $q_1 \sim_{\mathcal{A}} q_2 \sim_{\mathcal{A}} q'$ and $p_1 \sim_{\mathcal{A}} p_2$. On the right, a game where Eve can produce a word in $\mathcal{L}(\mathcal{A})$ but not positionally.

Prefix-independent coBüchi recognisable objectives Let us start by introducing some terminology relative to coBüchi automata. Our analysis requires considering automata that are not necessarily deterministic, however, we have a fine control of the non-determinism that will appear. First, all automata in this subsection will be history-deterministic. Moreover, we can suppose that they are deterministic over transitions producing priority 2 by the following result of Kuperberg et Skrzypczak [41] (this property is sometimes called safe determinism).

LEMMA 4.25 ([41]). *Every history-deterministic coBüchi automaton contains an equivalent subautomaton that is deterministic over 2-transitions, which can be computed in polynomial time.*

Safe languages and safe components. Consider a (possibly non-deterministic) coBüchi automaton $\mathcal{A}$. We define the (<2)-*safe language* of a state q (or just *safe language*) as the set of finite or infinite words such that, when read from q, priority 1 can be avoided, that is:

$$\mathsf{Safe}_{<2}(q) = \{w \in \Sigma^* \cup \Sigma^\omega \mid \text{ there is a run } q \xrightarrow{w:2} \}.$$

REMARK 4.26. Two safe languages coincide if and only if their restrictions to finite (resp. infinite) words coincide. Indeed, an infinite word $w \in \Sigma^\omega$ belongs to $\mathsf{Safe}_{<2}(q)$ if and only all its finite prefixes do.

We write $q \sqsubseteq_2 q'$ if $\mathsf{Safe}_{<2}(q) \subseteq \mathsf{Safe}_{<2}(q')$; this defines a partial preorder on states of $\mathcal{A}$. We will sometimes use the term *safe path* to refer to paths in $\mathcal{A}$ that do not produce priority 1. The use of the notation "(< 2)-safe" will be justified by the generalisation of this notion to any parity automaton. The next lemma follows directly from the definition of safe language.

LEMMA 4.27 (Monotonicity with respect to safe languages). *Let $\mathcal{A}$ be a coBüchi automaton which is deterministic over 2-transitions, and let q, q' be two states such that $q \sqsubseteq_2 q'$. Let u be a finite word in $\mathsf{Safe}_{<2}(q)$ and write $q \xrightarrow{u:2} p$. There is a unique path $q' \xrightarrow{u:2} p'$ and $p \sqsubseteq_2 p'$.*

A (<2)-*safe component* (or just *safe component*) of $\mathcal{A}$ is a set of states forming a strongly connected component in the subautomaton obtained by removing from $\mathcal{A}$ all transitions labelled 1. By Lemma 4.25, we can suppose that these subautomata are deterministic. Also, note that if $\mathcal{A}$ is in normal form, transitions between different safe components produce priority 1, that is, states connected by a safe path are in the same safe component.

Statement of the characterisation of positionality. We are now ready to state a characterisation of positionality of prefix-independent coBüchi recognisable objectives.

PROPOSITION 4.28 (Positionality for prefix-independent coBüchi recognisable objectives). *A prefix-independent coBüchi recognisable objective is positional if and only if it can be recognised by a deterministic coBüchi automaton satisfying that within each safe component, states are totally ordered by inclusion of safe languages.*

Before going on with the proof, we discuss two examples.

EXAMPLE 4.29. In Figure 11 we represent a coBüchi automaton over $\Sigma = \{a, b, c\}$ recognising the following objective:

$$W = \text{ Words containing finitely many factors in } c(a^*cb^*)^+c.$$

This is an example of an objective that is not concave, as by shuffling words $a(ccaa)^\omega \notin W$ and $(bbcc)^\omega \notin W$ we can obtain $(abcc)^\omega \in W$. However, we show that it satisfies the hypothesis of Proposition 4.28, so it is positional. This automaton has a single safe component. The inclusions of the safe languages follows from the fact that the transitions are monotone: for every letter $\alpha \in \Sigma$, if $q_i \xrightarrow{\alpha:2} q_j$ and $i \leq i'$, then $q_{i'} \xrightarrow{\alpha:2} q_{j'}$ with $j \leq j'$. ◆

EXAMPLE 4.30. Let $\Sigma = \{a, b, c\}$ and

$$W = \mathtt{Fin}(ac) \vee \mathtt{Fin}(bb).$$

We give a coBüchi automaton recognising W and satisfying the hypothesis of Proposition 4.28 in Figure 12. This automaton has two safe components: $S_1 = \{q_1, q_2\}$ and $S_2 = \{p_1, p_2\}$. The states of each component are totally ordered by inclusion of safe languages, as we have $q_1 \sqsubset_2 q_2$ and $p_1 \sqsubset_2 p_2$. Therefore, W is positional. ◆

REMARK 4.31. In his PhD thesis [34, Section 6.2], Kopczyński introduced a notion of *monotonic automata* over finite words, and showed that if $L \subseteq \Sigma^*$ is recognised by such an automaton, then

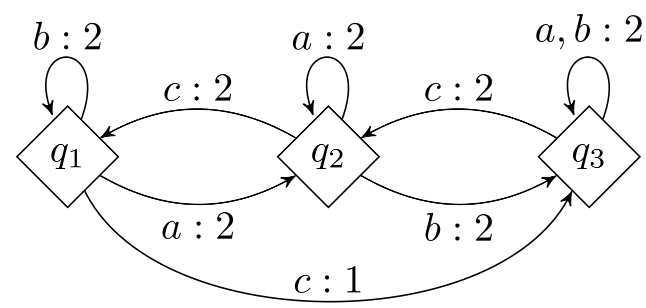

Figure 11. Deterministic coBüchi automaton recognising the objective W from Example 4.29.

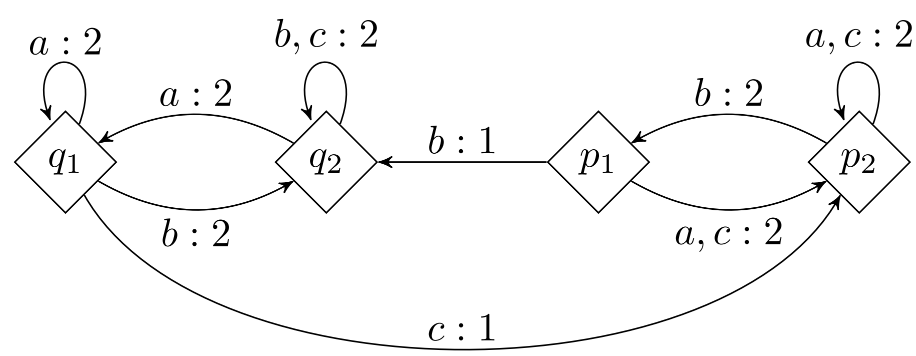

Figure 12. Deterministic coBüchi automaton recognising the objective W from Example 4.30. This automaton has two safe components: $S_1 = \{q_1, q_2\}$ and $S_2 = \{p_1, p_2\}$, and the states of each of them are totally ordered by inclusion of safe languages.

the (prefix-independent) objective $\Sigma^\omega \setminus L^\omega$ is positional [34, Prop 6.6]. It turns out that these correspond exactly to the objectives characterised in Proposition 4.28.

At the level of intuition, starting from a deterministic coBüchi automaton $\mathcal{A}$ recognising a positional objective W, our proof of the necessity in Proposition 4.28 proceeds as follows:

- We turn $\mathcal{A}$ into a history-deterministic safe centralised automaton (see definition below) using the minimisation technique of Abu Radi and Kupferman [1].
- Using positionality, we prove that $\sqsubseteq_2$ defines a total order on each safe component.
- Exploiting the total order, we are able to re-determinise $\mathcal{A}$.

Safe centralisation and safe minimality. Let $\mathcal{A}$ be a (possibly non-deterministic) coBüchi automaton with only one residual class. We say that $\mathcal{A}$ is *safe centralised* if, for every pair of states q_1, q_2, if $q_1 \sqsubseteq_2 q_2$, then q_1 and q_2 are in the same safe component. We say that $\mathcal{A}$ is *safe minimal* if there are no two different states with the same safe language.

The next lemma is a consequence of [1]. We present a self-contained proof that will be generalised in Lemma 5.16.

LEMMA 4.32. *Any prefix-independent coBüchi recognisable language can be recognised by a history-deterministic coBüchi automaton that is safe centralised and safe minimal.*

PROOF OF LEMMA 4.32. Let $\mathcal{A}$ be a normalised, history-deterministic, deterministic over 2-transitions automaton recognising W.

First, we show that we can assume $\mathcal{A}$ to be 1-*saturated*, that is, for all pairs of states q, q' and letters $a \in \Sigma$, the transition $q \xrightarrow{a:1} q'$ appears in $\mathcal{A}$.

CLAIM 4.33. *The coBüchi automaton obtained by adding all possible transitions of the form $q \xrightarrow{a:1} q'$ to $\mathcal{A}$ recognises W. Moreover, it is history-deterministic and deterministic over 2-transitions.*

Proof. Let $\mathcal{A}'$ be the automaton obtained adding all 1-transitions. To show that $\mathcal{L}(\mathcal{A}') \subseteq W$, we remark that an accepting run over a word w in $\mathcal{A}'$ eventually only reads transitions with priority 2, so it eventually coincides with a run in $\mathcal{A}$. We conclude by prefix-independence. For the other inclusion, and the fact that $\mathcal{A}'$ is history-deterministic, it suffices to use the same resolver as in $\mathcal{A}$. It is straightforward that $\mathcal{A}'$ is deterministic over 2-transitions. ◆

In the following, we assume that $\mathcal{A}$ is 1-saturated. We say that a safe component S is *redundant* if there is $q \in S$ and $q' \notin S$ such that $q \sqsubseteq_2 q'$.

CLAIM 4.34. *Let S be redundant and consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by deleting S. Then $\mathcal{A}'$ is history-deterministic and recognises W.*

Proof. Clearly $\mathcal{L}(\mathcal{A}') \subseteq W$. We will describe a sound resolver proving that $\mathcal{L}(\mathcal{A}') = W$ and that $\mathcal{A}'$ is history-deterministic. Let $q \in S$ and $q' \notin S$ such that $q \sqsubseteq_2 q'$. For each $p \in S$, pick $u \in \Sigma^*$ such that $q \overset{u:2}{\rightsquigarrow} p$, and let $f(p)$ be such that $q' \overset{u:2}{\rightsquigarrow} f(p)$; this is well defined since $q \sqsubseteq_2 q'$. Note that we have $p \sqsubseteq_2 f(p)$ by Lemma 4.27. By normality of $\mathcal{A}$, there is a returning path $f(p) \overset{w:2}{\rightsquigarrow} q'$ and thus $f(p)$ is in the same safe component as q', so it does not belong to S. We extend f to all of Q by setting it to be the identity over $Q \setminus S$.

Take a sound resolver (q_0, r) in $\mathcal{A}$, let $w \in \Sigma^\omega$, and write

$$\rho = q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \dots$$

for the run in $\mathcal{A}$ induced by r over w. We build a resolver (q'_0, r') in $\mathcal{A}'$ satisfying the property that the run induced over w, $\rho' = q'_0 \xrightarrow{w_0} q'_1 \xrightarrow{w_1} \dots$ is such that for each i, $q_i \sqsubseteq_2 q'_i$. We let $q'_0 = f(q_0)$, and assume ρ' constructed up to q'_i, and $q_i \sqsubseteq_2 q'_i$. If there is a state $q'_{i+1} \notin S$ such that $q'_i \xrightarrow{w_i:2} q'_{i+1}$ then we take this one, which satisfies $q_{i+1} \sqsubseteq_2 q'_{i+1}$ by Lemma 4.27. Otherwise, take the transition $q'_i \xrightarrow{w_i:1} f(q_{i+1})$ (which exists by 1-saturation). Therefore, if ρ is accepting, there is a suffix $w_i w_{i+1} \dots \in \mathsf{Safe}_{<2}(q_i) \subseteq \mathsf{Safe}_{<2}(q'_i)$, so the run $q'_i \xrightarrow{w_i:2} q'_{i+1} \xrightarrow{w_{i+1}:2} \dots$ in $\mathcal{A}'$ is safe, and ρ' is accepting too. ◆

Using Claim 4.34, we successively remove redundant safe components until obtaining a safe centralised automaton.

Finally, to obtain a safe minimal automaton it suffices to merge states with the same safe language. That is, we define a 1-saturated automaton that has for states the classes $[q]_2$ of states of $\mathcal{A}$, and transitions $[q]_2 \xrightarrow{a:2} [p]_2$ whenever for some (or equivalently, for all) state $q' \in [q]_2$ there is transition $q' \xrightarrow{a:2} p'$ in $\mathcal{A}$, with $p' \in [p]_2$. It is not difficult to check that the obtained automaton recognises W, is history-deterministic and remains safe centralised. ∎

Total order in each safe component. The intuitive idea on why having states of a same safe component totally ordered by $\sqsubseteq_2$ is necessary for positionality is the same than in the case of closed objectives (Lemma 4.1): if q and q' are incomparable, there are two words w, w' that produce priority 1 from one state but not from the other. In a game, if Eve has not kept track of where we are in the automaton, she will not know what is the best option between w and w'. However, an issue arises when turning this idea into an actual proof: one needs to build two full differentiating runs from q and q'; producing priority 1 just once does not suffice. Safe centrality will come in handy for this purpose.

By definition, if $q \not\sqsubseteq_2 q'$, there is a word which produces priority 1 when read from q', and stays in the corresponding safe component when read from q. The following lemma exploits safe centrality to extend those runs to synchronise them in the same state, while the run starting from q remains safe. For the purpose of the warm up, we only prove it assuming $\mathcal{A}$ is deterministic; extending it to the history-deterministic case requires some additional technicalities that will be dealt with in Section 5.2.

LEMMA 4.35 (Synchronisation of separating runs). *Let $\mathcal{A}$ be a normalised, safe centralised and safe minimal deterministic coBüchi automaton with a single residual class. Let q and q' be two states such that $q \not\sqsubseteq_2 q'$ and p be any state in the safe component of q. There is a word $w \in \Sigma^*$ such that $q \xrightarrow{w:2} p$ and $q' \xrightarrow{w:1} p$.*

PROOF. Since $\mathsf{Safe}_{<2}(q) \not\subseteq \mathsf{Safe}_{<2}(q')$, there is a word $w_1 \in \Sigma^*$ such that $q \xrightarrow{w_1:2} q_1$ and $q' \xrightarrow{w_1:1} q'_1$. By normality, note that q_1 is in the same safe component as q. If we have again that $q_1 \not\sqsubseteq_2 q'_1$, we can find a word w_2 with the same properties. While the non-inclusion of safe languages is satisfied, repeating the argument yields two runs:

$$q \xrightarrow{w_1:2} q_1 \xrightarrow{w_2:2} q_2 \xrightarrow{w_3:2} \dots$$
$$q' \xrightarrow{w_1:1} q'_1 \xrightarrow{w_2:1} q'_2 \xrightarrow{w_3:1} \dots$$

We claim that the process should stop after finite time, meaning that for some i, we have $q_i \sqsubseteq_2 q'_i$. Otherwise, we would obtain two infinite runs over $w_1 w_2 w_3 \dots \in \Sigma^\omega$, one of them accepting and the other rejecting, contradicting the fact that $\mathcal{A}$ has a single residual class.

Let i be the step in which $q_i \sqsubseteq_2 q'_i$. First, we note that both states are in the safe component of q: state q_i is in there because there is a path $q \xrightarrow{2} q_i$, and by safe centrality, q'_i must also be in the same safe component. Let $q_{\max}$ be a state in this safe component maximal amongst states

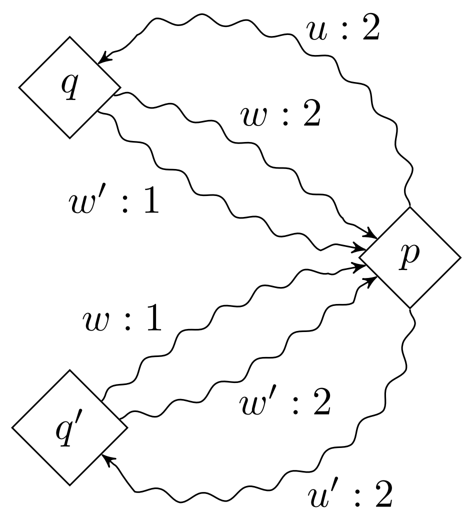

Figure 13. Situation occurring in the proof Lemma 4.36.

such that $q'_i \sqsubseteq_2 q_{\max}$. Let $u \in \Sigma^*$ be a word labelling a path $q_i \xrightarrow{u:2} q_{\max}$ (which exists by definition of safe components). By Lemma 4.27, maximality of $q_{\max}$, and safe minimality, we also have $q'_i \xrightarrow{u:2} q_{\max}$. Finally, if suffices to take a word $u' \in \Sigma^*$ labelling a path $q_{\max} \xrightarrow{u':2} p$ and define $w = w_1 w_2 \ldots w_i u u'$. ■

We may derive the sought total orders.

LEMMA 4.36 (Total order in each safe component). *Let $\mathcal{A}$ be a deterministic coBüchi automaton recognising a prefix-independent positional objective W. Suppose that $\mathcal{A}$ is safe centralised and safe minimal. Let q and q' be two different states in the same safe component. Then, either $q \sqsubset_2 q'$ or $q' \sqsubset_2 q$.*

PROOF. By safe minimality, $q \not\leq_2 q'$ implies $q \not\sqsubseteq_2 q'$. Suppose by contradiction that $q \not\sqsubseteq_2 q'$ and $q' \not\sqsubseteq_2 q$. Let p be a state in this safe component, and let $u, u' \in \Sigma^*$ be such that $p \xrightarrow{u:2} q$ and $p \xrightarrow{u':2} q'$. By Lemma 4.35, there are $w, w' \in \Sigma^\omega$ such that:

$$q \xrightarrow{w:2} p, \quad q \xrightarrow{w':1} p,$$
$$q' \xrightarrow{w:1} p, \quad q' \xrightarrow{w':2} p.$$

The situation is depicted in Figure 13. We obtain that:

— $(w'u)^\omega \notin W$,
— $(wu')^\omega \notin W$,
— $(wu'w'u)^\omega \in W$.

Thus consider the game in which Eve controls a vertex with two self loops labelled $w'u$ and wu'; she can win by alternating both loops, but fails to win positionally. ■

Determinisation. We have stated the last two lemmas of the previous paragraph for deterministic automata, in order to simplify the presentation. However, safe centralisation only yields history-deterministic automata. We now explain how to use the total order from Lemma 4.36 to determinise HD coBüchi automata recognising positional languages.

Let $\mathcal{A}$ be a normalised, history-deterministic and deterministic over 2-transitions coBüchi automaton recognising a prefix-independent positional language. Order $\sqsubseteq_2$ is total over each safe component, by Lemma 4.36. We show how to rearrange the 1-transitions in order to define an equivalent deterministic automaton $\mathcal{A}'$ with the same structure of safe components.

Let $\{S_1, S_2, \ldots, S_k\}$ be the safe components of $\mathcal{A}$, enumerated in an arbitrary order. If a word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$, there is a run over w that eventually stays in one of the components S_i. The main idea is that, when reading a word w, we can resolve the non-determinism by trying each safe component in a round-robin fashion. If for a state q and for a letter $a \in \Sigma$ there is a (unique) transition $q \xrightarrow{a:2}$, we keep it as the only a-transition from q. If there is no transition $q \xrightarrow{a:2}$, and q belongs to S_i, we define a transition $q \xrightarrow{a:1} q'$ towards some q' in S_{i+1}. The total order in S_{i+1} identifies a state in S_{i+1} which is the best to go to: we define $q \xrightarrow{a:1} q_{i+1}^{\max}$, where $q_{i+1}^{\max}$ is the unique maximal state of S_{i+1} for the total order $\sqsubseteq_2$. This defines a deterministic automaton $\mathcal{A}$'.

We prove that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. Clearly $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, as $\mathcal{A}'$ is a subautomaton of the 1-saturation of $\mathcal{A}$. To show the other inclusion, let $w \in \mathcal{L}(\mathcal{A})$. There is a run over w in $\mathcal{A}$ that eventually remains in a safe component, without loss of generality, we assume that it is S_1. Let $w' = w_1' w_2' \ldots$ be a suffix of w labelling a run in S_1: $q^1 \xrightarrow{w_1'} q^2 \xrightarrow{w_2'} \ldots$. Consider a run over w' in $\mathcal{A}'$. If this run never visits S_1, it must be because it eventually remains in some safe component, so w' is accepted by $\mathcal{A}'$ in that case. If the run eventually arrives to S_1 (at the i$^{\text{th}}$ step), it will arrive to the maximal element $q_1^{\max}$. At this point, the run in $\mathcal{A}$ is in $q^i \sqsubseteq_2 q_1^{\max}$. Since $w_i' w_{i+1}' \cdots \in \mathsf{Safe}_{<2}(q^i) \subseteq \mathsf{Safe}_{<2}(q_1^{\max})$, the run over this suffix is safe in $\mathcal{A}'$, and word w is accepted by $\mathcal{A}'$.

EXAMPLE 4.37. The automaton from Figure 12 has the shape we have described: transitions producing priority 1 cycle between the two safe components, and they go to the maximal state of the other component ($p_1 \xrightarrow{b:1} q_2$ and $q_1 \xrightarrow{c:1} p_2$). ◆

Generalisation to non prefix-independent coBüchi recognisable languages To remove the prefix-independence assumption, we work with the localisation to residuals of the objectives, as defined in Section 4.3. If W is a positional objective, for each residual $[u]$ the objective $W_{[u]}$ is positional. In turns out that this property, together with the hypothesis over residuals that were already necessary for open objectives, provides a characterisation.

PROPOSITION 4.38. *Let $W \subseteq \Sigma^\omega$ be a coBüchi recognisable language. Then, W is positional if and only if:*

- *$\mathsf{Res}(W)$ is totally ordered,*
- *W is progress consistent, and*
- *for all residual class $[u]$, objective $W_{[u]}$ is positional.*

This result is not fully satisfying (and hard to prove directly), as it relies on the positionality of languages $W_{[u]}$. As these objectives are prefix-independent, we have a characterisation for their positionality (Proposition 4.28), and we can put them together to obtain a statement using exclusively structural properties of parity automata. The statement we obtain uses a hyerarchical decomposition of parity automata in three layers:

— States are totally preordered by their residual class (layer 0).
— Within each residual class, states are grouped into safe components (layer 1).
— Within each safe component, states are totally ordered by inclusion of the safe languages (layer 2).

This hyerarchical decomposition foreshadows the definition of structured signature automata that we will use in Section 5.2 to derive a characterisation of positionality for all ω-regular languages.

PROPOSITION 4.39. *Let $W \subseteq \Sigma^\omega$ be a coBüchi recognisable language. Then, W is positional if and only if:*

— $\mathsf{Res}(W)$ *is totally ordered,*
— *W is progress consistent, and*
— *W can be recognised by a deterministic coBüchi automaton $\mathcal{A}$ such that, for all residual class $[q]$, the local automaton of the residual $\mathcal{A}_{[q]}$ satisfies that its safe components are totally ordered by inclusion of safe languages.*

We do not include a proof of this proposition; it is a special case of Theorem 3.1.

EXAMPLE 4.40. We consider the following objective over the alphabet $\Sigma = \{a, b, c\}$:

$$W = \Sigma^* a^\omega \cup \Sigma^* b^\omega \cup c\Sigma^* c\Sigma^\omega.$$

This objective was studied in [3, Lemma 12] to show that there are positional objectives that are not concave, nor bipositional (objectives from Examples 4.18 and 4.29 also have this property); their proof of positionality is quite involved.

A coBüchi automaton recognising the objective W is depicted in Figure 14. Its residuals are totally ordered, and it is easy to check that it is progress consistent. Moreover, all its safe components are trivial. Therefore, Proposition 4.39 implies that it is positional. ◆

4.5 Towards objectives of higher complexity: An example

We revisit the example from Figure 4, depicted here in Figure 15. It recognises the objective of words that either contain 'a' infinitely often, or contain no 'a' at all and only finitely many occurrences of the factor 'bb':

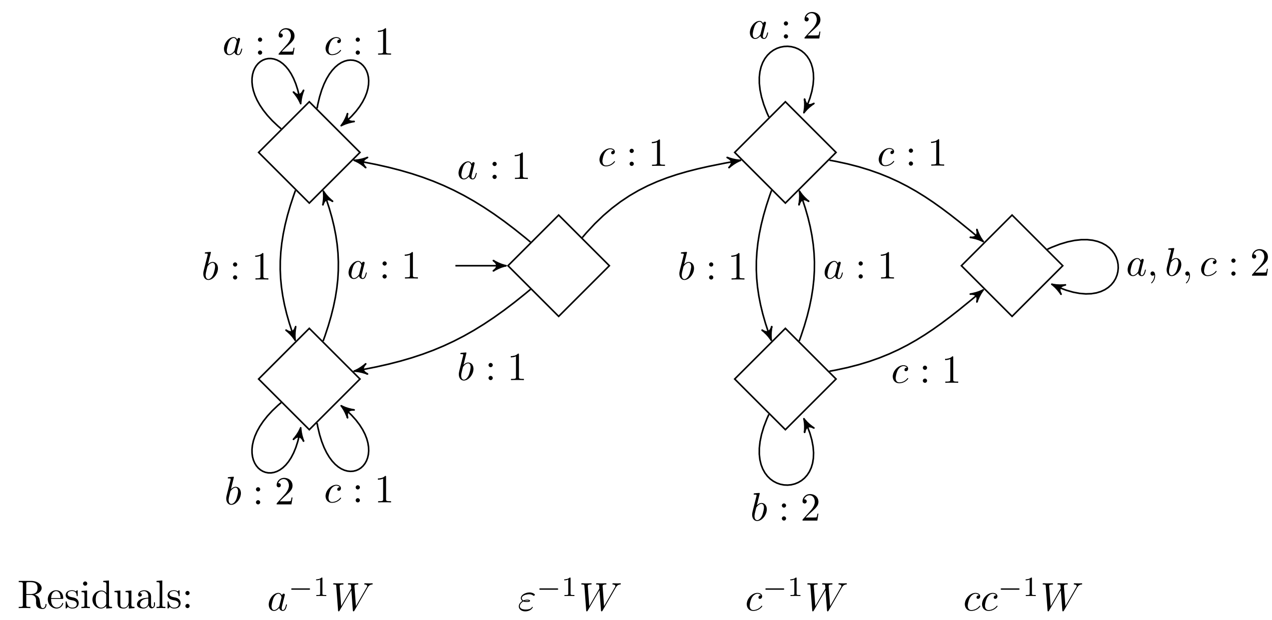

Figure 14. Automaton recognising the objective W from Example 4.40.

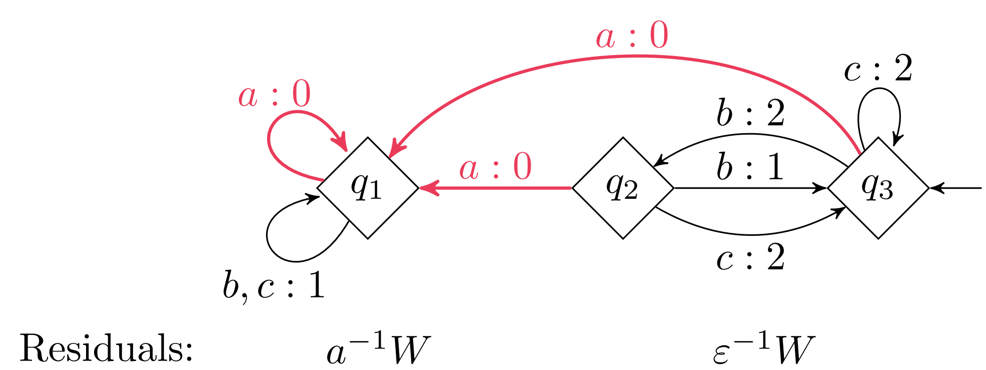

Figure 15. Automaton recognising the objective $W = \texttt{Inf}(a) \vee (\texttt{Fin}(bb) \wedge \texttt{No}(a))$.

$$W = \texttt{Inf}(a) \vee (\texttt{No}(a) \wedge \texttt{Fin}(bb)), \quad \text{over } \Sigma = \{a, b, c\}.$$

This objective is neither Büchi nor coBüchi recognisable, so none of the characterisations of this section applies to it. However, we can combine the techniques presented above to equip $\mathcal{A}$ with a "nicely behaved" total order. In the next section, we will see that this can be formalised as the fact that $\mathcal{A}$ is a deterministic fully progress consistent signature automaton, so by Theorem 3.1, W is positional.

The objective W has two residuals: $\varepsilon^{-1}W$ and $x^{-1}W$. It is clear that $a^{-1}W \subseteq \varepsilon^{-1}W$, and, since there is no transition $[a] \to [\varepsilon]$, it is trivially progress consistent. The states associated to $[a]$ and $[\varepsilon]$ are, respectively, $\{q_1\}$ and $\{q_2, q_3\}$. The subautomaton over $\{q_1\}$ recognises the Büchi objective $\texttt{Buchi}(a)$, which is positional. Let us focus on the subautomaton induced by $\{q_2, q_3\}$, that coincides with $\mathcal{A}_{[\varepsilon]}$. We observe that it satisfies the hypothesis of Lemma 4.19: transitions with priority 0 act uniformly within $\mathcal{A}_{[\varepsilon]}$; indeed, these transitions are those reading letter a (in red). Consider the restriction of this subautomaton to letters $\{b, c\}$. The obtained automaton $\mathcal{A}'_{[\varepsilon]}$ is a coBüchi automaton, satisfying the hypothesis of Lemma 4.36: the states of the safe component of $\mathcal{A}'_{[\varepsilon]}$ are totally ordered by inclusion of safe languages, as $q_2 \sqsubset_2 q_3$. In this way, we obtain a hyerarchical decomposition such that:

- Residuals are totally ordered by inclusion, and are progress consistent.
- 0-transitions act uniformly within the states of each residual class.
- The safe components of the coBüchi automaton obtained as the restriction of each class to transitions with priority ≥ 1 are totally ordered by inclusion of safe languages.

Generalising such hyerarchical decomposition to any parity automaton will be the central point of the next section.

We also note that the ε-completion of this automaton presented in Figure 4 follows the decomposition presented here: $\xrightarrow{\varepsilon:0}$-transitions follow the order of the residuals, and $\xrightarrow{2}$-transitions the one given by the inclusion of safe languages. This procedure of ε-completion will be generalised in Section 5.3.

5. Obtaining the structural characterisation of positionality

We now move on to the proof of Theorem 3.1. The first step is to identify the common structural properties of deterministic parity automata recognising positional objectives. In Section 5.1, we define a class of parity automata, which we call signature automata, that underscores this structure. We also introduce full progress consistency, a further necessary condition for automata recognising positional languages. To obtain the implication from (1) to (2) in Theorem 3.1 we need then to show how to obtain a fully progress consistent signature automaton for a positional ω-regular objective. This is the most technical part of the proof, and it is the object of Section 5.2. We then proceed to defining ε-complete automata and closing the cycle of implications in Section 5.3.

5.1 Signature automata and full progress consistency

We first introduce more precise concepts about congruences.

5.1.1 Priority-faithful congruences and quotient automata

Priority-faithful congruences. We recall that a congruence in an automaton allows us to define a quotient $\mathcal{A}/_\sim$, which is a deterministic automaton structure. However, in general, no colouring with priorities can be defined on top of $\mathcal{A}/_\sim$ in a sensible way, as the congruence does not have to be compatible with the output priorities of the automaton. We now strengthen the definition of congruence for parity automata so that it will be possible to define an approximation of a correct parity condition on top of the quotient automaton.

DEFINITION 5.1 ($[0,x]$-faithful congruence). Let $\mathcal{A}$ be a parity automaton and let $\sim$ be a congruence over its set of states Q. We say that $\sim$ is $[0,x]$-*faithful* if:

- for each $0 \leq y \leq x$, y-transitions are uniform over $\sim$-classes and relation $\sim$ is a congruence for y-transitions, and
- relation $\sim$ is a congruence for $(>x)$-transitions.

Stated differently, a relation $\sim$ is a $[0,x]$-faithful congruence if, whenever there is a transition $q \xrightarrow{a:y} p$ with priority $y \leq x$, then for every $q' \sim q$ a transition $q' \xrightarrow{a} p'$ produces priority y and goes to a state $p' \sim p$. If there is a transition $q \xrightarrow{a:y} p$ producing priority $y > x$, we only require that transitions $q' \xrightarrow{a:>x} p'$ produce priorities $> x$ and go to $p' \sim p$ (but the exact priority produced may differ). (We remark that, although it is not explicitly imposed, $(>x)$-transitions are uniform over $\sim$-classes by the first property. Also, we recall that we assume that automata are complete.)

REMARK 5.2 ($[0,x]$-faithful congruences for deterministic automata). For deterministic automata we can give a simpler definition. An equivalence relation $\sim$ on a deterministic automaton $\mathcal{A}$ is a $[0,x]$-faithful congruence if and only if, whenever $q \sim q'$, $q \xrightarrow{a:y} p$ and $q' \xrightarrow{a:y'} p'$, then $p \sim p'$ and $y = y'$ if $y \leq x$.

REMARK 5.3. A $[0,x]$-faithful congruence is $[0,y]$-faithful for any $y \leq x$.

$(\leq x)$-quotient automaton. What the definition of $[0,x]$-faithful congruence tells us is that transitions producing priorities $y \leq x$ are well defined in the quotient automaton $\mathcal{A}/_\sim$, in the sense that we can associate a priority y to these transitions reliably. For transitions producing priorities $> x$, on the other hand, we only obtain the information that the priority produced from any state of the class will be large, but we lose some precision.

DEFINITION 5.4 ($(\leq x)$-quotient automaton). If $\sim$ is a $[0,x]$-faithful congruence, we can define the $(\leq x)$-*quotient of $\mathcal{A}$ by* $\sim$ to be the parity automaton $\mathcal{A}/_{\sim}^{\leq x}$ given by:

- Its set of states are the $\sim$-classes of $\mathcal{A}$.
- The initial state is $[q_{\mathsf{init}}]$, where q_{init} is the initial state of $\mathcal{A}$.
- For $y \leq x$, there is a transition $[q] \xrightarrow{a:y} [p]$ if $\mathcal{A}$ has a transition $q' \xrightarrow{a:y} p'$ with $q' \in [q]$, $p' \in [p]$.
- There is a transition $[q] \xrightarrow{a:x'} [p]$ if $\mathcal{A}$ has a transition $q' \xrightarrow{a:>x} p'$ with $q' \in [q]$, $p' \in [p]$; where $x' = x + 1$ if x even, and $x' = x$ if x odd.

The automaton $\mathcal{A}/_{\sim}^{\leq x}$ is defined so transitions coming from those producing a priority $> x$ in $\mathcal{A}$ are assigned the smallest odd priority not smaller than x. This guarantees that the projection of runs that eventually only produce priorities $> x$ in $\mathcal{A}$ are rejecting in $\mathcal{A}/_{\sim}^{\leq x}$. The next lemma refines this comment.

LEMMA 5.5. *Let $\mathcal{A}$ be a parity automaton and let $\sim$ be a $[0,x]$-faithful congruence over it. The $(\le x)$-quotient of $\mathcal{A}$ by $\sim$ recognises the language:*

$$\mathcal{L}(\mathcal{A}/^{\sim}_{\le x}) = \{w \in \Sigma^\omega \mid w \textit{ is accepted with an even priority } y \le x \textit{ in } \mathcal{A}\}.$$

Moreover, if $\mathcal{A}$ is in normal form, then so is $\mathcal{A}/^{\sim}_{\le x}$.

PROOF. A run ρ in $\mathcal{A}$ produces a priority $y \le x$ infinitely often if and only its projection in $\mathcal{A}/^{\sim}_{\le x}$ produces priority y infinitely often, which gives the equality of the languages. It is a direct check that $\mathcal{A}/^{\sim}_{\le x}$ inherits being in normal form. ∎

5.1.2 Signature automata

We give the definition of signature automata, at the core of our characterisation.

We say that a sequence of total preorders $\sqsubseteq_0, \sqsubseteq_1, \sqsubseteq_2, \dots, \sqsubseteq_k$ over Q is a *collection of nested total preorders* if $\sqsubseteq_i$ refines $\sqsubseteq_{i-1}$, for $i > 0$. We note that, in that case, the induced equivalence relation $\sim_i$ also refines $\sim_{i-1}$.

DEFINITION 5.6 (Signature automaton). Let $d \in \mathbb{N}$ be a priority. A *d-signature automaton* is a semantically deterministic parity automaton $\mathcal{A}$ together with a collection of nested total preorders $\sqsubseteq_0, \sqsubseteq_1, \sqsubseteq_2, \dots, \sqsubseteq_d$ over Q such that:[7]

I) **Refinements of residual inclusion.** Preorder $\sqsubseteq_0$ refines the preorder $\sqsubseteq_{\mathcal{A}}$ given by the inclusion of residuals.

II) **Faithful partitions at even layers.** For $0 \le x \le d$, x even, the equivalence relation $\sim_x$ is a $[0,x]$-faithful congruence.

III) **$(<x)$-*safe separation* at odd layers.** For $2 \le x \le d$, x even, and $q \sim_{x-2} q'$:

$$q \sqsubset_{x-1} q' \quad \Longrightarrow \quad \text{there is no path } q \xrightarrow{w:\ge x} q'.$$

IV) ***Local monotonicity* of $(\ge x)$-transitions.** For an even priority $x \le d$, transitions using priorities $\ge x$ are monotone for $\sqsubseteq_x$ over each $\sim_{x-1}$ class. That is, for $q \sim_{x-1} q'$, if $q \sqsubseteq_x q'$:

$$q \xrightarrow{a:\ge x} p \quad \Longrightarrow \quad q' \xrightarrow{a:\ge x} p',\ p \sim_{x-1} p' \text{ and } p \sqsubseteq_x p',\ \text{ for all } a\text{-transitions from } q'.$$

We say that $\mathcal{A}$ is a *signature automaton* if it is a d-signature automaton, for d the maximal priority appearing in $\mathcal{A}$.

We note that even and odd preorders play a completely different role in the previous definition. In fact, the only purpose of odd preorders is to delimit the areas in which the local monotonicity property will apply. Item (III) constrains $\sim_{x-1}$-classes to be "sufficiently large".

7 For notational convenience, we let $\sim_{-2}$ be the complete relation over $\mathcal{A}$ throughout this definition. That is, $q \sim_{-2} p$ for all pairs of states in Q.

REMARK 5.7. We note that, by Item (IV), for x even the equivalence relations $\sim_{x-1}$ is a congruence for $\geq x$-transitions. However, the restrictions on these odd preorders are much weaker, as we do not impose them to be faithful.

EXAMPLE 5.8. Consider the automaton $\mathcal{A}$ from Figure 15 from the warm-up. This automaton has 3 states, q_1, q_2, and q_3. It can be equipped with the structure of a signature automaton as follows:

- Preorder $\sqsubseteq_0$ is given by the inclusion of residuals: $q_1 \sqsubset_0 q_2, q_3$, and $q_2 \sim_0 q_3$.
- Preorder $\sqsubseteq_1$ coincides with preorder $\sqsubseteq_0$.
- Preorder $\sqsubseteq_2$ is a total order: $q_1 \sqsubset_2 q_2 \sqsubset_2 q_3$.

◆

Signature automata are not minimal in general, but we conjecture that by merging $\sim_d$-equivalent states we should obtain a minimal automaton (see Section 9.2 for more discussions).

5.1.3 Full progress consistency.

The existence of a signature automaton recognising an objective W does not suffice to ensure positionality of W. The problem is similar to the one we encountered when studying open objectives in Section 4.2: there are open objectives whose residuals are totally ordered but they are not positional (see Example 4.4). In that case, we needed to add the property of progress consistency to characterise positionality. We generalise this notion to signature automata with multiple preorders.

DEFINITION 5.9 (Full progress consistency). We say that a signature automaton $\mathcal{A}$ is *fully progress consistent* if, for each preorder $\sqsubseteq_x$, for x even, and every finite word $w \in \Sigma^*$:

$$q \sqsubset_x p \;\text{ and }\; q \xrightarrow{w:\geq x} p \quad \Longrightarrow \quad w^\omega \in \mathcal{L}(\mathcal{A}_q).$$

REMARK 5.10. A fully progress consistent signature automaton is in particular progress consistent, as the $\sqsubseteq_0$-preorder refines the preorder coming from the inclusion of residuals.

5.1.4 Structured signature automata from semantic properties of languages

To prove the implication (1) $\Longrightarrow$ (2) from Theorem 3.1, we build a signature automaton from a deterministic parity automaton $\mathcal{A}$ recognising W recursively. In order to be able to carry out the recursion, we will in fact obtain a signature automaton with even stronger properties. This reinforcement of signature automata is done by ensuring that the preorders $\sqsubseteq_x$ come from semantic properties of the automaton, for which the notion of $<x$-safe languages will play a major role. The properties that are imposed are essentially a generalisation of the ones satisfied by the canonical history-deterministic coBüchi automaton defined by Abu Radi and Kupferman [1].

We introduce some further notation used in our semantic reinforcement of the definition of a signature automaton.

$<x$-safe languages. Let $\mathcal{A}$ be a (possibly non-deterministic) parity automaton. We define the $(<x)$-*safe language* of a state q of $\mathcal{A}$ as:

$$\mathsf{Safe}^{\mathcal{A}}_{<x}(q) = \{w \in \Sigma^* \cup \Sigma^\omega \mid \text{ there exists } q \overset{w:\geq x}{\rightsquigarrow}\}.$$

We remark that $\mathsf{Safe}^{\mathcal{A}}_{<x}(q)$ is completely determined by its finite (resp. infinite) words. We drop the superscript $\mathcal{A}$ whenever the automaton is clear from the context. A path producing no priority strictly smaller than x is called $(<x)$-*safe*.

Next lemma simply follows from the definition.

LEMMA 5.11 (Monotonicity of safe languages). *Let $\mathcal{A}$ be a parity automaton that is deterministic over transitions using priorities $\geq x$. Let q and p be two states such that $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)$, and let $q \overset{u:\geq x}{\rightsquigarrow} q'$ be a $<x$-safe run over u from q. Then, there is a unique $<x$-safe run over u from p, $p \overset{u:\geq x}{\rightsquigarrow} p'$, and it leads to a state p' satisfying $\mathsf{Safe}_{<x}(q') \subseteq \mathsf{Safe}_{<x}(p')$.*

$<x$-safe components. A $(<x)$-*safe component* of $\mathcal{A}$ is a strongly connected component of the subautomaton obtained by removing all transitions producing a priority $< x$ from $\mathcal{A}$. Note that if $\mathcal{A}$ is in normal form and $x > 0$, transitions changing of $(<x)$-safe component produce a priority $< x$. That is, $q \overset{u:\geq x}{\rightsquigarrow} p$ implies that q and p are in the same $(<x)$-safe component.

REMARK 5.12. The partition of $\mathcal{A}$ into $<x$-safe components is a refinement of its partition into $<y$-safe components, for $y \leq x$.

For the following, we fix a parity automaton $\mathcal{A}$ in normal form using priorities in $[0, d_{\max}]$. For each $x \in [1, d_{\max}]$, we will totally order the $<x$-safe components of $\mathcal{A}$ in such a way that these orders successively refine each other. For $x \in [1, d_{\max}]$ we let $S_1^{<x}, \dots, S_{k_x}^{<x}$ be the $<x$-safe components of $\mathcal{A}$. For $x = 1$, we let $S_1^{<1} <_1 S_2^{<1} <_1 \cdots <_1 S_{k_1}^{<1}$ be an arbitrary order over the <1-safe components. Assume that an order has already been fixed at level $x - 2$. Then, we fix an arbitrary total order for the $<x$-safe components contained in a same $<(x-1)$-safe components, which yields a total order for the set of all those safe components, that refines the previous layers. From now on, we assume that the enumerations $S_1^{<x}, \dots, S_{k_x}^{<x}$ correspond to these orders: $S_i^{<x} <_x S_j^{<x}$ if $i < j$.

Structured signature automata. The preorders of the signature automaton we plan to build will correspond to the following semantic properties:

1. **Preorder 0 given by inclusion of residuals.** Preorder $\sqsubseteq_0$ corresponds to the inclusion of residuals:

$$q \sqsubseteq_0 p \iff \mathcal{L}(\mathcal{A}_q) \subseteq \mathcal{L}(\mathcal{A}_p).$$

2. **Odd layers correspond to safe components.** For $x \geq 2$ even, we define $\sqsubseteq_{x-1}$ by:

$$q \sqsubseteq_{x-1} p \iff q \sqsubset_{x-2} p \text{ or } [q \sim_{x-2} p \text{ and } q \in S_i^{<x} \text{ and } p \in S_j^{<x} \text{ with } i \leq j].$$

   In particular, $q \sim_{x-1} p$ if and only if $q \sim_{x-2} p$ and there is a path $q \xrightarrow{w:\geq x} p$.
3. **Even preorders given by inclusion of safe languages.** For $x \geq 2$, x even, we define $\sqsubseteq_x$ by:

$$q \sqsubseteq_x p \iff q \sqsubset_{x-1} p \text{ or } [q \sim_{x-1} p \text{ and } \mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)].$$

These preorders already ensure some of the properties required to be a signature automaton; mainly, the local monotonicity of transitions using large priorities (Item (IV)), as well as the congruence for $\geq x$-transitions at $\sim_x$-classes. Note, however, that it is not clear (and will be an important part of our proof) that $\sqsubseteq_x$ is total for even x.

Given an (even or odd) priority $d \in \mathbb{N}$, we say that a parity automaton $\mathcal{A}$ in normal form together with nested preorders $\sqsubseteq_0, \sqsubseteq_1, \ldots, \sqsubseteq_d$ as above is a *d-structured signature automaton* if these preorders are total and moreover:

4. **Strong congruence of $(\leq x)$-priorities over even classes.** Let $0 \leq x \leq d$, x even. For every $y \leq x$:

$$q \sim_x q',\ q \xrightarrow{a:y} p \text{ and } q' \xrightarrow{a:z} p' \implies z = y \text{ and } p = p'.$$

5. **Classes at layer x are $(>x)$-connected.** For $0 \leq x \leq d$ and $q \sim_x q'$, we have:

$$q \neq q' \implies \text{there is a path } q \xrightarrow{u:>x} q'.$$ [8]

6. **Safe centralisation.** Let $2 \leq x \leq d$ be an even priority, and let $q \sim_{x-2} p$. Then:

$$q \not\sim_{x-1} p \implies \mathsf{Safe}_{<x}(q) \not\subseteq \mathsf{Safe}_{<x}(p).$$

We say that $\mathcal{A}$ is a *structured signature automaton* if it is a d-structured signature automaton, for d the maximal priority appearing in $\mathcal{A}$.

REMARK 5.13. We draw the reader's attention to the fact that in Item 4 we do not only require $p \sim_x p'$, but impose $p = p'$. This will be necessary to guarantee that the relations $\sim_y$ for even priorities $y > x$ are also congruences for x-transitions.

LEMMA 5.14. *A deterministic d-structured signature automaton is a d-signature automaton.*

PROOF. The fact that $\sqsubseteq_0$ refines the inclusion of residuals is ensured by Item 1. Also, the $(<x)$-safe separation at odd levels (Item (III)) is directly implied by the fact that odd layers correspond to safe components (Item 2).

8 We remark that, for x odd, this property is already implied by Item 2.

We now show by induction on x that for each $x \leq d$, x even, $\sim_x$ is a $[0,x]$-faithful congruence. Consider two states $q \sim_x q'$, which rewrites as $q \sim_{x-1} q'$ and $\mathsf{Safe}_{<x}(q) = \mathsf{Safe}_{<x}(p)$, and pick a transition $q \xrightarrow{a:y} p$. There are two cases.

— If $y \leq x$, then by Item 4, we have $q' \xrightarrow{a:y} p$.

— If $y > x$, then by monotonicity of safe languages (Lemma 5.11), we have $q' \xrightarrow{a:\geq x} p'$ with $\mathsf{Safe}_{<x}(p') = \mathsf{Safe}_{<x}(p)$, and by induction, $p' \sim_{x-2} p$. From Item 6, it follows that $p' \sim_{x-1} p$ and thus $p' \sim_x p$, as required.

We conclude that $\sim_x$ is a $[0,x]$-faithful congruence.

Finally, the local monotonicity of $(\geq x)$-transitions follows from the fact that even preorders correspond to the inclusion of safe languages (Item 3) and the monotonicity of safe languages (Lemma 5.11). ■

5.2 From positionality to signature automata

This section is devoted to the proof of the implication (1) $\implies$ (2) in Theorem 3.1. Many of the ideas in this proof have already appeared in the warm-up section. However further technical issues stem from the fact that we manipulate general parity automata. Details for a number of proofs are relegated to Appendix A.

GLOBAL HYPOTHESIS. In the whole section, W stands for an objective that is positional over finite, ε-free Eve-games. These hypotheses will not necessarily be recalled in the statements of propositions.

5.2.1 Outline of the induction

Given a deterministic parity automaton recognising a positional objective, we will recursively define the preorders and equivalence relations making $\mathcal{A}$ a structured signature automaton. The base case consists in showing that the preorder $\sqsubseteq_0$ given by the inclusion of residuals is total, and ensuring Item 4 of the definition for this preorder. For the recursion step, we suppose that we have a deterministic $(x-2)$-structured signature automaton $\mathcal{A}$ recognising W, for x even, and we define preorders $\sqsubseteq_{x-1}$ and $\sqsubseteq_x$ over $\mathcal{A}$ as imposed by Items 2 and 3. Then, we apply a sequence of operations, after which we obtain an equivalent deterministic automaton, that is either x-structured signature, or has strictly less states than $\mathcal{A}$. In the first case, we continue to define preorders $\sqsubseteq_{x+1}$ and $\sqsubseteq_{x+2}$; in the second case, we restart the structuration procedure from the beginning, with a strictly smaller automaton. In both cases, we conclude by induction.

We conjecture that we can sequentially obtain all the preorders, without having to restart the construction at each step. However, we have not been able to overcome some technical difficulties preventing us to do so. We refer to the final subsection of Appendix A for more details.

We give a more detailed account on the specific operations we apply to obtain the different items of the definition of a structured signature automaton and their order:

i) **Relation $\sim_{x-1}$ and safe centralisation.** We define $\sim_{x-1}$, as determined by Item 2. Applying a generalisation of the procedure from [1], we $(<x)$-safe centralise $\mathcal{A}$, obtaining an equivalent automaton satisfying Item 6. The resulting automaton is no longer deterministic, but it is history-deterministic and has a very restricted and controlled amount of non-determinism.

ii) **Total order in safe components.** We prove that the states of each $(<x)$-safe component are totally ordered by inclusion of $(<x)$-safe languages (for which we rely on the safe centralisation hypothesis). This shows that the preorder $\sqsubseteq_x$ given by the inclusion of safe languages (Item 3) is total.

iii) **Re-determinisation.** We determinise automaton $\mathcal{A}$, while preserving previously obtained properties. For this, the fact that $\sqsubseteq_x$ is total will be key.

iv) **Uniformity of x-transitions.** Finally, we show that either $\mathcal{A}$ already satisfies Items 4 and 5, or we can trim the automaton to an equivalent strictly smaller one.

Moreover, we show that all these transformations can be performed in polynomial time.

This establishes that an objective W that is positional over finite, ε-free Eve-games can be recognised by a deterministic structured signature automaton. At the end of the section, we show that such an automaton must be fully progress consistent (Lemma 5.23).

5.2.2 Constructing structured signature automata for positional languages

Let $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, [0, d_{\max}], \Delta, \mathsf{parity})$ be a deterministic parity automaton recognising W, and suppose that W is positional over finite, ε-free Eve-games. We assume that $\mathcal{A}$ is in normal form.

In this subsection, we will apply successive transformations to the automaton $\mathcal{A}$, ensuring an increasing list of properties. At the beginning of each paragraph, we clearly state the properties that are assumed. We allow ourselves to omit these hypotheses in the statements of propositions inside the paragraphs.

Base case: Preorder $\sqsubseteq_0$. We define $q \sqsubseteq_0 p$ if $\mathcal{L}(\mathcal{A}_q) \subseteq \mathcal{L}(\mathcal{A}_p)$, as imposed by Item 1. In Lemma 4.1, we showed that positionality of W implies that this order is total. However, in our proof we used infinite, and not necessarily ε-free games. It is not difficult to modify the proof to adapt to this set of minimal hypotheses, using ω-regularity of W. We give all details in Appendix A (Lemma A.9). Items 2, 3, and 6 are trivially satisfied. Therefore, it suffices to show that we can obtain an automaton such that $\sim_0$ is a strong congruence for transitions producing priority 0 (Item 4), and that $\sim_0$-equivalent states can be connected by paths producing priority $>x$ (Item 5). For this, we apply exactly the same method presented in Section 4.3: we obtain a polished automaton and show that it satisfies the desired properties. This proof will be covered in the recursive step; the case $x = 0$ does not present any particularity.

Moving on to the inductive step, for the rest of the subsection, we let x be an even priority such that $2 < x \leq d_{\max}$ and assume that $\mathcal{A}$ is a deterministic $(x-2)$-structured signature automaton.

Safe centrality and relation $\sim_{x-1}$ We say that an automaton with a preorder $\sqsubseteq_{x-2}$ is $(<x)$-*safe centralised* if $\sim_{x-2}$-equivalent states that are comparable for the inclusion of $(<x)$-safe languages are in the same $(<x)$-safe component.

REMARK 5.15. For automata in normal form $(<x)$-safe centrality can be stated as: if $q \sim_{x-2} p$ and there is no $(<x)$-safe path connecting q and p, then $\mathsf{Safe}_{<x}(q) \nsubseteq \mathsf{Safe}_{<x}(p)$.

LEMMA 5.16 ($(<x)$-safe centralisation). *There exists a $(x-2)$-structured signature automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ which is:*

- *deterministic over transitions with priority different from $x-1$,*
- *homogeneous,*
- *history-deterministic, and*
- *$(<x)$-safe centralised.*

Moreover, $\mathcal{A}'$ can be obtained in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

The proof of this lemma is a refinement of the corresponding result for coBüchi automata presented in the warm-up (Lemma 4.32): we saturate $\sim_{x-2}$-classes of the original automaton $\mathcal{A}$ with $(x-1)$-transitions, and then remove redundant $(<x)$-safe components recursively until obtaining a $(<x)$-safe centralised automaton. We include all details in Appendix A (page 90).

Lemma 5.16 allows us to define $\sim_{x-1}$ satisfying all required properties: for $q \sim_{x-2} p$, we define $q \sqsubseteq_{x-1} p$ if and only if $q \in S_i^{<x}$ and $p \in S_j^{<x}$ with $i \leq j$, where $S_i^{<x}$ are the $(<x)$-safe components of $\mathcal{A}$ enumerated following the order described in Section 5.1. By definition, Item 2 is satisfied, and by $(<x)$-safe centralisation of $\mathcal{A}$, so is Item 6.

Preorder $\sqsubseteq_x$: Total order given by safe languages In all this paragraph we assume that $\mathcal{A}$ is an automaton as obtained in the previous paragraph, that is: it has nested preorders defined up to $\sqsubseteq_{x-1}$ making it a $(x-2)$-structured signature automaton and satisfying Items 2 and 6 for relation $\sim_{x-1}$. Moreover, it is history-deterministic, homogeneous, and the only non-determinism of $\mathcal{A}$ appears in $(x-1)$-transitions.

We define preorder $\sqsubseteq_x$ as imposed by Item 3:

$$q \sqsubseteq_x p \iff q \sqsubset_{x-1} p \text{ or } [q \sim_{x-1} p \text{ and } \mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)],$$

and recall that it follows that $q \sim_x p$ if and only if $q \sim_{x-1} p$ and there is a $(<x)$-safe path from q to p.

REMARK 5.17. Using Item 4 for priorities $y \leq x-2$ and Lemma 5.11 for transitions with priority $\geq x$, we get that relation $\sim_x$ is a congruence for transitions with a priority different from $x-1$. Moreover, over each $\sim_{x-1}$-class, transitions with priority $\geq x$ are monotone for $\sqsubseteq_x$.

Our objective is now to show that $\sqsubseteq_x$ is total over each $\sim_{x-1}$-class. The proof of this statement uses the same ideas as the corresponding result from the warm-up (Lemma 4.36). In particular, the main technical point resides in proving that, for two states $q_1 \not\sqsubseteq_x q_2$, we can force to produce priority $x-1$ from q_1 while remaining $<x$-safe from q_2, and then resynchronise both paths in a same $\sim_x$-class. This result, stated in Lemma 5.19, is the analogue to Lemma 4.35 from the warm-up. For its proof we strongly rely on the $(<x)$-safe centralisation of $\mathcal{A}$ and the fine control of its non-determinism.

In the proofs of Lemmas 5.19 and 5.20 we will reason at the level of $\sim_x$-classes. As we only suppose that $\mathcal{A}$ is $(x-2)$-structured, we do not have the uniformity of x-transitions over $\sim_x$-classes yet. Lemma 5.18 below provides a weaker version of this uniformity that will suffice for the arguments in the upcoming lemmas.

We say that a word w *produces priority y uniformly* in a class $[q]_x$ if for every $q' \in [q]_x$ all runs from q' are of the form $q' \xrightarrow{w:y}$. In that case, we write $[q]_x \xrightarrow{w:y}$. We say that such a word produces priority x uniformly in $[q]_x$ *leading to* $[p]_x$ if for every $q' \in [q]_x$ we have $q' \xrightarrow{w:y} p'$ with $p' \in [p]_x$. In that case, we write $[q]_x \xrightarrow{w:y} [p]_x$.

We note that whenever $\mathcal{A}$ contains a path $q \xrightarrow{w:\geq x} p$, a run over w is unique, as $\mathcal{A}$ is homogeneous and its restriction to transitions coloured with priorities $\geq x$ is deterministic.

The proof of the next lemma combines normality of $\mathcal{A}$ with ideas appearing in the proof of Claim 4.14 from the warm-up; all the details can be found in Appendix A (page 103).

LEMMA 5.18 (Existence of uniform words). *Let p and q be two states from the same $(<x)$-safe component. There is a word $w \in \Sigma^*$ producing priority x uniformly in $[q]_x$ leading to $[p]_x$.*

We next state the result that allows us to synchronise runs in a same $\sim_x$-class. Its proof is analogous to that of Lemma 4.35 and can be found in Appendix A.

We let r be a sound resolver for $\mathcal{A}$, and assume that all states can be reached by a run induced by this resolver. We recall that we write $q \xrightarrow[\forall,\mathsf{r}]{w:y} p$ if, for every word $u_0 \in \Sigma^*$ such that the induced run of r over u_0 arrives to q, the induced run of r over $u_0 w$ ends in p and produces y as minimal priority in the part of the run corresponding to w. Recall also that we write $[q]_x \xrightarrow[\forall,\mathsf{r}]{w:y} [p]_x$ if for any $q' \in [q]_x$ we have $q' \xrightarrow[\forall,\mathsf{r}]{w:y} p'$ for some $p' \in [p]_x$.

LEMMA 5.19 (Synchronisation of separating runs). *Suppose that $q \sim_{x-1} q'$ and $q \not\sqsubseteq_x q'$ and let $p \in [q]_{x-1}$. There is a word $w \in \Sigma^+$ such that $[q]_x \xrightarrow[\forall,\mathsf{r}]{w:x-1} [p]_x$ and $[q']_x \xrightarrow[\forall,\mathsf{r}]{w:x} [p]_x$.*

We can now deduce that $\sqsubseteq_x$ is total over each $\sim_{x-1}$-class.

LEMMA 5.20 (Total order in $(<x)$-safe components). *Let $q, q' \in Q$ be two states such that $q \sim_{x-1} q'$. Then, either $q \sqsubseteq_x q'$ or $q' \sqsubseteq_x q$.*

PROOF. Suppose by contradiction that $\mathsf{Safe}_{<x}(q) \not\subseteq \mathsf{Safe}_{<x}(q')$ and $\mathsf{Safe}_{<x}(q') \not\subseteq \mathsf{Safe}_{<x}(q)$. Let p be a state in $[q]_{x-1} = [q']_{x-1}$, and apply Lemma 5.18 to obtain words $u, u' \in \Sigma^*$ such that $[p]_x \xrightarrow{u:x} [q]_x$ and $[p]_x \xrightarrow{u':x} [q']_x$.

By Lemma 5.19, there are words $w, w' \in \Sigma^\omega$ such that:

$$[q]_x \xrightarrow[\forall,\mathsf{r}]{w:x} [p]_x, \qquad [q]_x \xrightarrow[\forall,\mathsf{r}]{w':x-1} [p]_x,$$
$$[q']_x \xrightarrow[\forall,\mathsf{r}]{w:x-1} [p]_x, \qquad [q']_x \xrightarrow[\forall,\mathsf{r}]{w':x} [p]_x.$$

The situation is analogous to the one depicted in Figure 13 in the warm-up. We obtain that:

- $(w'u)^\omega \notin \mathcal{L}(\mathcal{A}_q)$,
- $(wu')^\omega \notin \mathcal{L}(\mathcal{A}_q)$,
- $(wu'w'u)^\omega \in \mathcal{L}(\mathcal{A}_q)$.

Let $u_0 \in \Sigma^*$ be a word such that the run induced by r over u_0 ends in q (it exists, as we have supposed that all states are reachable using r). It suffices to consider the game where there is path labelled u_0 leading to a vertex controlled by Eve with two self loops; one of them producing $w'u$ and the other wu'. By the previous remarks, she can win such game by alternating both loops, but she cannot win positionally. ■

Re-obtaining determinism In this paragraph we assume that $\mathcal{A}$ is a parity automaton recognising W equipped with nested total preorders defined up to $\sqsubseteq_x$ with all properties obtained until now:

- it is a $(x-2)$-structured signature automaton,
- preorder $\sqsubseteq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton,
- preorder $\sqsubseteq_x$ satisfies the property from Item 3 from the definition of a structured signature automaton,
- it is deterministic over transitions with priorities different from $x-1$,
- it is homogeneous, and
- it is history-deterministic.

We claim that we can obtain a deterministic equivalent automaton preserving the entire structure of total preorders. Moreover, in the obtained automaton we guarantee that relation $\sim_x$ satisfies Item 4 from the definition of a structured signature automaton for priorities $y < x$.

LEMMA 5.21 (Re-determinisation). *There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ with nested total preorders defined up to $\sqsubseteq_x$ satisfying that:*

- *it is a $(x-2)$-structured signature automaton,*

— *preorder $\sqsubseteq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and*
— *preorder $\sqsubseteq_x$ is a congruence and satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4 .*

Moreover, automaton $\mathcal{A}'$ can be computed in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

The idea of the proof is a direct generalisation of the one presented in the warm-up for coBüchi automata (page 39): we redefine the $(x-1)$-transitions of the automaton in such a way that we ensure that a run that changes of $<x$-safe component infinitely often passes through all these components in a round-robin fashion. The total order $\sqsubseteq_x$ allows us to identify a maximal state in each component, so we can make a deterministic choice. Formal details can be found in Appendix A (page 99).

Uniformity of x-transitions over $\sim_x$-classes We assume that $\mathcal{A}$ is a deterministic parity automaton recognising W with nested total preorders defined up to $\sqsubseteq_x$ satisfying all conditions stated in Lemma 5.21. The objective of this paragraph is to obtain the remaining properties of a x-structured signature automaton (Items 4 and 5).

LEMMA 5.22 (Uniformity of x-transitions over $\sim_x$-classes). *There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ such that either:*
— *$\mathcal{A}'$ is an x-structured signature automaton with $|\mathcal{A}'| \leq |\mathcal{A}|$, or*
— *$|\mathcal{A}'| < |\mathcal{A}|$.*

In both cases, such an automaton can be computed in polynomial time from $\mathcal{A}$.

The proof of this lemma generalises the techniques introduced in Section 4.3 of the warm-up. Details can be found in Appendix A (from page 103). We introduce the local automaton of a $\sim_x$-class $[q]_x$: the automaton originated by keeping the states of $[q]_x$ and paths connecting them producing priorities $\geq x$. Using positionality and ideas analogous to those from Lemma 4.13, we show that these local automata admit a well-defined set of super letters, that is, there are letters that, if read infinitely often in such a local automaton, must produce an accepting word. These letters are exactly the ones carrying priority x when read from $[q]_x$ in the final automaton $\mathcal{A}'$.

To obtain the uniformity of x-transitions, we might need to simplify the automaton: we introduce x-polished automata, the target form of automata that will allow us to obtain uniformity of x-transitions. Using the existence of super letters, we show that we can polish automaton $\mathcal{A}$ by removing redundant parts of it. This operation might break the normal form of automaton $\mathcal{A}$,[9] but this is not a problem, since in any case it strictly decreases the number of states of the automaton, as desired.

9 In fact, we believe that the polishing operation does preserve normality, but we have not been able to prove it.

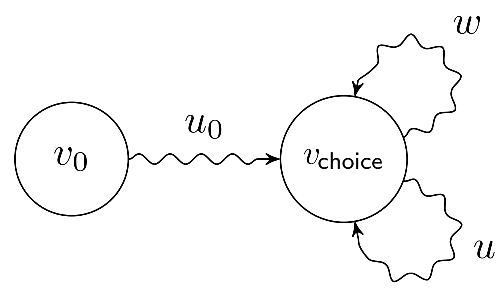

Figure 16. A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if $\mathcal{A}$ is not fully progress consistent, as in the proof of Lemma 5.23.

This ends the induction step of the proof, establishing existence of a deterministic structured signature automaton recognising W.

5.2.3 Full progress consistency

We show that a structured signature automaton recognising a positional objective must be fully progress consistent. Since we showed how to obtain such a structured signature automaton in the previous section, this ends the proof of the implication (1) $\implies$ (2) from Theorem 3.1.

LEMMA 5.23 (Necessity of full progress consistency). *Let $W \subseteq \Sigma^\omega$ be positional over finite, ε-free Eve-games. Any structured signature automaton recognising W is fully progress consistent.*

PROOF. Suppose by contradiction that $\mathcal{A}$ is a structured signature automaton for W that is not fully progress consistent. By definition, for some priority x even, there are $q \sqsubset_x p$ and a word $w \in \Sigma^*$ such that $q \xrightarrow{w:\geq x} p$, but $w^\omega \notin \mathcal{L}(\mathcal{A}_q)$. As $\sim_x$ is a $[0, x]$-faithful congruence, we can work with $\sim_x$-classes and write $[q]_x \xrightarrow{w:\geq x} [p]_x$. We study first the case $x > 0$. By Lemma 5.19, there is a word $u \in \Sigma^+$ such that $[q]_x \xrightarrow{u:x-1} [q]_x$ and $[p]_x \xrightarrow{u:x} [q]_x$. Let $u_0 \in \Sigma^+$ be a word reaching q from the initial state of $\mathcal{A}$.[10] We obtain:

- $u_0 w^\omega \notin W$,
- $u_0 u^\omega \notin W$, and
- $u_0 (wu)^\omega \in W$.

We consider the game depicted in Figure 16. Eve can win from v_0 by alternating loops labelled w and u when the play arrives to v_{choice}. However, she cannot win positionally from v_0.

For the case $x = 0$, the proof is almost identical to that of Lemma 4.6; we just need to ensure that the game in Figure 7 (page 26) can be supposed finite and ε-free. Finiteness of the game can be obtained by using ultimately periodic words. To guarantee that we do not include ε-transitions, if $u_0 = \varepsilon$, we remove vertex v_0 from the game. ■

10 If q is initial, we omit u_0 and the state v_0 of the game from Figure 16 to ensure the use of an ε-free game.

5.3 From signature automata to positionality through ε-complete automata

We now complete the equivalence of the statements from Theorem 3.1, excluding (3”), by showing the implications (2) $\implies$ (3) $\implies$ (3’) $\implies$ (4). The implication (4) $\implies$ (5) follows from Proposition 2.2 (taken from [47]) and (5) $\implies$ (1) is trivial.

5.3.1 ε-complete automata

We start with our crucial definition.

DEFINITION 5.24. An *ε-complete* automaton $\mathcal{A}$ is a non-deterministic parity automaton (with ε-transitions) with priorities ranging between 0 and $d+1$, where d is even, such that

- the relations $\xrightarrow{\varepsilon:1}, \xrightarrow{\varepsilon:3}, \dots, \xrightarrow{\varepsilon,d+1}$ all define total preorders, each refining the previous one;
- for each even $x \in \{0, 2, \dots, d\}$, the relation $\xrightarrow{\varepsilon:x}$ is the strict variant of $\xrightarrow{\varepsilon:x+1}$: for any q, q', it holds that $q \xrightarrow{\varepsilon:x} q'$ if and only if $q' \xrightarrow{\varepsilon:x+1} q$ does not hold.

We say that an automaton $\mathcal{A}$ (which will usually be taken deterministic) is *ε-completable*, if one may add ε-transitions to $\mathcal{A}$ so as to make it ε-complete, without augmenting the language. We say that the resulting (generally non-deterministic) automaton $\mathcal{A}'$ is an *ε-completion* of $\mathcal{A}$; note that if $\mathcal{A}$ is deterministic, then $\mathcal{A}'$ is history-deterministic (it is even determinisable by pruning). This provides the implication from (3) to (3’) in Theorem 3.1 We refer to Figure 4 in Section 3 for an example.

5.3.2 From signature automata to ε-completable automata

We now prove the implication (2) $\implies$ (3) from Theorem 3.1, which can be stated as follows.

LEMMA 5.25 (From (2) to (3) in Theorem 3.1). *Let $\mathcal{A}$ be a fully progress consistent deterministic signature automaton. Then $\mathcal{A}$ is ε-completable.*

We prove Lemma 5.25. We refer to the discussion at the end of Section 4.5 (Figure 15) for an example on the ideas of this proof. Let $\mathcal{A}$ be a fully progress consistent deterministic signature automaton with nested preorders $\sqsubseteq_0, \sqsubseteq_1, \dots, \sqsubseteq_d$ and let $W = \mathcal{L}(\mathcal{A})$. Consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by adding, for all even priorities $x \in [0, d]$, transitions $q \xrightarrow{\varepsilon:x+1} q'$ whenever $q' \sqsubseteq_x q$ and $q \xrightarrow{\varepsilon:x} q'$ whenever $q' \sqsubset_x q$. Note that $\mathcal{A}'$ (potentially) has transitions with priorities up to $d+1$.

REMARK 5.26. Note that, for x even, $q \xrightarrow{\varepsilon:\geq x} p$ in $\mathcal{A}'$ entails $p \sqsubseteq_x q$.

Since by definition, $q' \sqsubset_x q$ is the negation of $q \sqsubseteq_x q'$, it follows immediately that $\mathcal{A}'$ is ε-complete. Moreover, as $\mathcal{A}$ is a subautomaton of $\mathcal{A}'$, the inclusion $W \subseteq \mathcal{L}(\mathcal{A}')$ is trivial. The difficulty lies in showing that $\mathcal{L}(\mathcal{A}') \subseteq W$.

REMARK 5.27. If $q \overset{w:x}{\rightsquigarrow} q$ is a cycle in $\mathcal{A}'$ producing an even minimal priority, then w is not composed exclusively of ε-letters.

For a priority x (even or odd), we say that a transition $q \xrightarrow{\varepsilon} q'$ in $\mathcal{A}'$ is an *x-jump* if $q' \sqsubset_x q$. We remark that if $x' \leq x$, an x'-jump is an x-jump. We start with a useful technical lemma.

LEMMA 5.28. *Fix a path $q' \overset{w':\geq x}{\rightsquigarrow} p'$ in $\mathcal{A}'$, with x even, and consider a run $q \overset{w}{\rightsquigarrow} p$ in $\mathcal{A}$, where w is obtained from w' by removing ε-letters (where $p = q$ if w is empty).*

a) *Assume that there is no x-jump on $q' \overset{w':\geq x}{\rightsquigarrow} p'$ and that $q \sim_x q'$. Then $p \sim_x p'$ and $q \overset{w:\geq x}{\rightsquigarrow} p$ in $\mathcal{A}$. Moreover, if $q' \overset{w':x}{\rightsquigarrow} p'$, then $q \overset{w:x}{\rightsquigarrow} p$ in $\mathcal{A}$.*

b) *Assume that there is no $(x-1)$-jump on $q' \overset{w':\geq x}{\rightsquigarrow} p'$, that $q' \sim_{x-1} q$ and $q' \sqsubseteq_x q$. Then $p' \sim_{x-1} p$, $p' \sqsubseteq_x p$ and $q \overset{w:\geq x}{\rightsquigarrow} p$ in $\mathcal{A}$.*

c) *We have that $p' \sqsubseteq_{x-1} q'$.*

PROOF. In the two first cases we deal with the case of a letter and conclude by induction.

a) There are two possibilities, depending on whether the letter is ε or not.
   - Transition $q' \xrightarrow{a:\geq x} p'$ with $a \in \Sigma$. Then $[0, x]$-faithfulness of $\sim_x$ gives $p \sim_x p'$ and $q \xrightarrow{a:\geq x} p$. Moreover if $q' \xrightarrow{a:x} p'$, then by $[0, x]$-faithfulness, $q \xrightarrow{a:x} p$.
   - Transition $q' \xrightarrow{\varepsilon:\geq x} p'$. Then $p' \sqsubseteq_x q'$ and since there is no x-jump, $p' \sim_x q'$. Thus $p = q \sim_x q' \sim_x p'$.

b) We distinguish the two same cases.
   - Transition $q' \xrightarrow{a:\geq x} p'$ with $a \in \Sigma$. Then local monotonicity of $(\geq x)$-transitions in $\mathcal{A}$ yields $q \xrightarrow{a:\geq x} p$ in $\mathcal{A}$ with $p' \sqsubseteq_x p$. By Remark 5.7, $p' \sim_{x-1} p$.
   - Transition $q' \xrightarrow{\varepsilon:\geq x} p'$. This implies $p' \sqsubseteq_x q'$ and since there is no $(x-1)$-jump, we have $p' \sim_{x-1} q'$. Thus we conclude that $p' \sqsubseteq_x q' \sqsubseteq_x q = p$ and $p = q \sim_{x-1} q' \sim_{x-1} p'$.

c) Suppose by contradiction that $p' \sqsupset_{x-1} q'$, and let q'_1 be the first state in the run such that $q' \sqsubset_{x-1} q'_1$. We have:

$$q' \overset{w'_1:\geq x}{\rightsquigarrow} q'_2 \xrightarrow{a:\geq x} q'_1 \overset{w'_2}{\rightsquigarrow} p',$$

with $q'_2 \sqsubset_{x-1} q'_1$. As in particular $q'_2 \sqsubset_x q'_1$, $a \neq \varepsilon$ (Remark 5.26). However, this contradicts Item (III) from the definition of signature automaton. ■

We now state the key result for deriving Lemma 5.25.

LEMMA 5.29. *Consider a cycle $q' \overset{w':x}{\rightsquigarrow} q'$ in $\mathcal{A}'$ with x even, and let w be obtained from w' by removing ε-letters. Then, w^ω is accepted from q' in $\mathcal{A}$.*

PROOF. We note that by Remark 5.27, w is not empty. Let y be minimal such that an y-jump appears on the path $q' \overset{w':x}{\rightsquigarrow} q'$ (and $y = d + 1$ if no y-jump occurs).

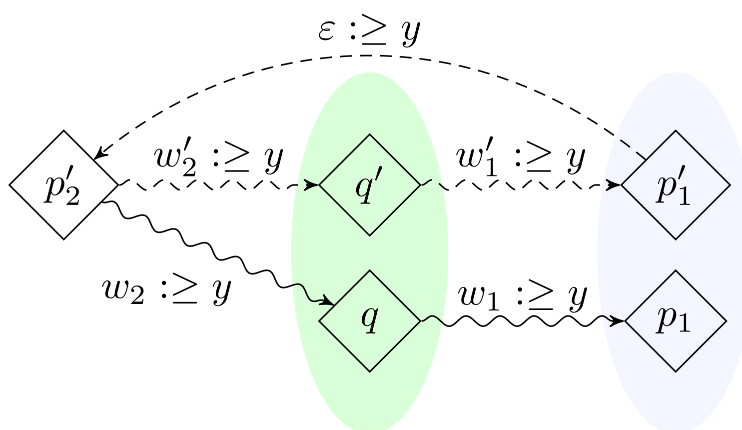

Figure 17. Situation in the third case of Lemma 5.29. Dashed lines represent paths in $\mathcal{A}'$ and solid lines those in $\mathcal{A}$. States that are $\sim_y$-equivalent are encircled together.

— If $y \geq x$. Then there is no $\xrightarrow{\varepsilon:x}$ transition on the path $q' \overset{w':x}{\rightsquigarrow} q'$ (otherwise it would produce an x-jump). Thus Lemma 5.28.a proves $q' \overset{w:x}{\rightsquigarrow} q_1$ in $\mathcal{A}$ with $q_1 \sim_x q'$. Then, since the $\sim_x$-class is preserved, successive applications of Lemma 5.28.a give $q' \overset{w:x}{\rightsquigarrow} q_1 \overset{w:x}{\rightsquigarrow} q_2 \overset{w:x}{\rightsquigarrow} q_3 \overset{w:x}{\rightsquigarrow} \dots$ in $\mathcal{A}$, and thus w^ω is accepted from q' in $\mathcal{A}$.

— If $y < x$ and y odd. We show that this case cannot happen. Let $p'_1 \xrightarrow{\varepsilon} p'_2$ denote the first y-jump on the path $q' \overset{w}{\rightsquigarrow} q'$ in $\mathcal{A}'$, that is, we have

$$q' \overset{w'_1:>y}{\rightsquigarrow} p'_1 \xrightarrow{\varepsilon:>y} p'_2 \overset{w'_2:>y}{\rightsquigarrow} q' \text{ in } \mathcal{A}', \quad p'_2 \sqsubset_y p'_1.$$

By Lemma 5.28.c, we have that $p'_1 \sqsubseteq_y q'$, so $p'_2 \sqsubset_y q'$. The existence of a path $p'_2 \overset{w'_2:\geq y}{\rightsquigarrow} q'$ contradicts Lemma 5.28.c.

— If $y < x$ and y even. Let $p'_1 \xrightarrow{\varepsilon} p'_2$ denote the last y-jump on the path $q' \overset{w}{\rightsquigarrow} q'$ in $\mathcal{A}'$, that is, we have

$$q' \overset{w'_1:\geq y}{\rightsquigarrow} p'_1 \xrightarrow{\varepsilon:\geq y} p'_2 \overset{w'_2:\geq y}{\rightsquigarrow} q' \text{ in } \mathcal{A}', \quad p'_2 \sqsubset_y p'_1,$$

and there is no y-jump on $p'_2 \overset{w_2}{\rightsquigarrow} q'$. We let w_1, w_2 be obtained, respectively, from w'_1 and w'_2 by removing ε's. By Lemma 5.28.a, we get that $p'_2 \overset{w_2:\geq y}{\rightsquigarrow} q$ in $\mathcal{A}$ for some $q \sim_y q'$. As there is no $(y-1)$-jump in the path, by Lemma 5.28.b, we get that $q \overset{w_1:\geq y}{\rightsquigarrow} p_1$ in $\mathcal{A}$ for $p'_2 \sqsubset_y p'_1 \sqsubseteq_y p_1$. See Figure 17 for an illustration of the situation.

All in all, we have obtained a path

$$p'_2 \overset{w_2:\geq y}{\rightsquigarrow} q \overset{w_1:\geq y}{\rightsquigarrow} p_1 \sqsupset_y p'_2 \quad \text{in } \mathcal{A}.$$

Therefore, full progress consistency yields $(w_2 w_1)^\omega \in \mathcal{L}(\mathcal{A}_{p'_2})$. As $\mathcal{L}(\mathcal{A}_{p'_2}) \subseteq \mathcal{L}(\mathcal{A}_{q'}) = \mathcal{L}(\mathcal{A}_q)$, we conclude that $w^\omega = (w_1 w_2)^\omega \in \mathcal{L}(\mathcal{A}_q)$. ■

We are now ready conclude the proof of Lemma 5.25.

PROOF OF LEMMA 5.25. As mentioned above, the inclusion $W \subseteq \mathcal{L}(\mathcal{A}')$ is trivial, as $\mathcal{A}$ is a subautomaton of $\mathcal{A}'$. This shows that, if the converse inclusion holds, $\mathcal{A}'$ is determinisable by pruning and therefore it is also history-deterministic.

We show $\mathcal{L}(\mathcal{A}') \subseteq W$. Take an accepting run in $\mathcal{A}'$ over $w' \in \Sigma^\omega$ and decompose it as:

$$q_0 \xrightarrow{w'_0} q' \xrightarrow{w'_1:x} q' \xrightarrow{w'_2:x} q' \xrightarrow{w'_3:x} \dots,$$

where x is even. For each i, let w_i be obtained from w'_i by removing ε's (which is non-empty by Remark 5.27), and consider the corresponding run in $\mathcal{A}$:

$$q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} q_2 \xrightarrow{w_2} q_3 \xrightarrow{w_3} \dots,$$

It follows by induction that $q' \sqsubseteq_0 q_i$, so, as order $\sqsubseteq_0$ refines the order of residuals (property (I) of a signature automaton), words that are accepted from q' in $\mathcal{A}$ are also accepted from q_i. By Lemma 5.29, it holds that for each pair of indices $j \leq j'$ we have $(w_j w_{j+1} \dots w_{j'})^\omega \in (q')^{-1}W$, so these words are also accepted from q_i, for all i.

Let $i_1, i_2, \dots$ be a sequence of indices such that $q_{i_j} = q_{i_{j+1}}$ for all j, and let $\tilde{q} = q_{i_1}$ be such recurring state. Each word $w_{i_j} \dots w_{i_j - 1}$ forms a cycle over $\tilde{q}$, that, by the previous remark, must be accepting, so the minimal priority produced on it is even. Therefore, we have found a decomposition of the run over w in $\mathcal{A}$ of the form

$$q_0 \xrightarrow{w_0 w_1 \dots} \tilde{q} \xrightarrow{w_{i_1} \dots w_{i_2 - 1} : x_1} \tilde{q} \xrightarrow{w_{i_2} \dots w_{i_3 - 1} : x_2} \dots,$$

with all x_i even. We conclude that $w_0 w_1 \dots \in W$. ■

5.3.3 Universal graphs from ε-complete automata

As noted at the end of Subsection 5.3.1, the implication (3) $\implies$ (3') in Theorem 3.1 is immediate. We now move on to implication (3') $\implies$ (4), stated as follows:

PROPOSITION 5.30 ((3') $\implies$ (4) in Theorem 3.1). *If there exists an ε-complete history-deterministic automaton recognising W, then there exists a well-ordered monotone (κ, W)-universal graph for each cardinal κ.*

For the rest of the section, we let $\mathcal{A}$ be an ε-complete history-deterministic automaton recognising W, and we let d be even such that $\mathcal{A}$ has priorities up to $d+1$, as in the above section.

Closure of an ε-complete automaton We define the order $\preccurlyeq$ over priorities in $[0, d+1]$ that sets $y \preccurlyeq x$ if x is "preferable" to y, that is: $1 \preccurlyeq 3 \preccurlyeq \dots \preccurlyeq d+1 \preccurlyeq d \preccurlyeq \dots \preccurlyeq 2 \preccurlyeq 0$.

REMARK 5.31. For any pair of infinite words $w, w' \in [0, d+1]^\omega$ satisfying that for all i $w_i \preccurlyeq w'_i$, it holds that:

$$w \in \mathsf{parity}_{[0,d+1]} \implies w' \in \mathsf{parity}_{[0,d+1]}.$$

We say that an automaton $\mathcal{A}$ is *priority-closed* if:

- for any states q, q', priorities $y' \preccurlyeq y$, and $a \in \Sigma \cup \{\varepsilon\}$

$$q \xrightarrow{a:y} q' \implies q \xrightarrow{a:y'} q'$$

- for any states p, p', q, q' and $a \in \Sigma \cup \{\varepsilon\}$,

$$p \xrightarrow{\varepsilon:y_1} q \xrightarrow{a:y_2} q' \xrightarrow{\varepsilon:y_3} p' \implies p \xrightarrow{a:\min_{\preccurlyeq}(y_1,y_2,y_3)} p'.$$

It is easy to turn any automaton into a priority-closed one.

LEMMA 5.32. *Let $\mathcal{A}$ be an automaton recognising W. There is an automaton $\mathcal{A}'$ recognising W which is priority-closed. Moreover, if $\mathcal{A}$ is history-deterministic and ε-complete, then so is $\mathcal{A}'$.*

PROOF. Let $\mathcal{A}'$ be obtained by adding to $\mathcal{A}$ all transitions of the form $q \xrightarrow{a:y'} q'$, when $q \xrightarrow{a:y} q'$ is a transition in $\mathcal{A}$ and $y' \preccurlyeq y$, and all transitions of the form $p \xrightarrow{a:\min_{\preccurlyeq}(y_1,y_2,y_3)} p'$, whenever a path $p \xrightarrow{\varepsilon:y_1} q \xrightarrow{a:y_2} q' \xrightarrow{\varepsilon:y_3} p'$ appears in $\mathcal{A}$. Clearly, $\mathcal{A}'$ is priority-closed, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ and if $\mathcal{A}$ is ε-complete, then so is $\mathcal{A}'$. The fact that this operation preserves history-determinism is also clear, once the equality of languages is obtained. To prove that $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, take an accepting run over $w \in \Sigma^\omega$ in $\mathcal{A}'$. We build a run over w in $\mathcal{A}$ by replacing any newly added transition $q \xrightarrow{a:y'} q'$ by $q \xrightarrow{a:y} q'$, and $p \xrightarrow{a:\min_{\preccurlyeq}(y_1,y_2,y_3)} p'$ by $p \xrightarrow{\varepsilon:y_1} q \xrightarrow{a:y_2} q' \xrightarrow{\varepsilon:y_3} p'$, respectively. By Remark 5.31, the obtained run is accepting in $\mathcal{A}$. ∎

In a priority-closed automaton, for each priority y, transitions $\xrightarrow{\varepsilon:y}$ define a transitive relation. If the automaton is moreover ε-complete, then for each even priority x, transitions of the form $\xrightarrow{\varepsilon:x+1}$ define total preorders.

We define $q \geq_x q'$ if $q \xrightarrow{\varepsilon:x+1} q'$. Note that, since $\mathcal{A}$ is priority-closed, these preorders are nested: $q \geq_{x+2} q'$ implies $q \geq_x q'$. Moreover, since $\mathcal{A}$ is ε-complete, for any even x:

$$q >_x q' \implies q \xrightarrow{\varepsilon:x} q'.$$

Finally, observe that, by priority-closure, states q, q' that are $\leq_d$-equivalent have exactly the same incoming and outgoing transitions, and can thus be merged without altering the language (this transformation preserves history-determinism). Therefore, we may assume that $\leq_d$ is antisymmetric and thus defines a total order on Q.

We write $[q]_x$ to denote the equivalence class of q associated to the preorder $\leq_x$. That is, $[q]_x$ contains the states q' such that $q \leq_x q'$ and $q' \leq_x q$.

Definition of the graph For the remainder of the section, we fix a cardinal κ. Let us first recall the construction of the (κ, parity)-universal graph U_{parity} for the parity objective over $\{0, \dots, d+1\}$ (see Example 2.4 for a proof of universality). Its vertices are of the form $(\lambda_1, \lambda_3, \dots, \lambda_{d+1}) \in \kappa^{d/2+1}$, ordered lexicographically, and its edges are given by

$$(\lambda_1, \dots, \lambda_{d+1}) \xrightarrow{x} (\lambda'_1, \dots, \lambda'_{d+1}) \iff \begin{cases} (\lambda'_1, \dots, \lambda'_{x-1}) \leq (\lambda_1, \dots, \lambda_{x-1}), & \text{if } x \text{ is even,} \\ (\lambda'_1, \dots, \lambda'_x) < (\lambda_1, \dots, \lambda_x), & \text{otherwise.} \end{cases}$$

Fix a priority-closed ε-complete and history-deterministic automaton $\mathcal{A}$ with states Q such that $\leq_d$ defines a total order on Q.

We define a Σ-graph $U_{\mathcal{A}}$ as follows. Vertices of $U_{\mathcal{A}}$ are the tuples $v = (q, \lambda_1, \lambda_3, \dots, \lambda_{d+1}) \in Q \times \kappa^{d/2+1}$. We associate to each such vertex the extended tuple

$$\text{ext}(v) = ([q]_0, \lambda_1, [q]_2, \lambda_3, \dots, [q]_{d-1}, \lambda_{d+1}).$$

We use it to define the total order: $v \leq v'$ if $\text{ext}(v)$ is smaller than $\text{ext}(v')$ for the lexicographic order. This is therefore a well-order. Edges in $U_{\mathcal{A}}$ are given by:

$$(q, \lambda_1, \dots, \lambda_{d+1}) \xrightarrow{a} (q', \lambda'_1, \dots, \lambda'_{d+1}) \iff \exists y \begin{cases} q \xrightarrow{a:y} q' \text{ in } \mathcal{A}, \quad \text{and} \\ (\lambda_1, \dots, \lambda_{d+1}) \xrightarrow{y} (\lambda'_1, \dots, \lambda'_{d+1}) \text{ in } U_{\text{parity}}. \end{cases}$$

Paths in $U_{\mathcal{A}}$ are well-behaved with respect to W, as stated below.

LEMMA 5.33. *Let $(q, \lambda_1, \dots, \lambda_{d+1}) \overset{w}{\rightsquigarrow}$ be an infinite path in $U_{\mathcal{A}}$. Then, $w \in q^{-1}W$.*

PROOF. Consider a path

$$(q^0, \lambda^0_1, \dots, \lambda^0_{d+1}) \xrightarrow{w_0} (q^1, \lambda^1_1, \dots, \lambda^1_{d+1}) \xrightarrow{w_1} \dots \text{ in } U_{\mathcal{A}}.$$

By definition, there are priorities $y_0, y_1, \dots$ such that

$$q^0 \xrightarrow{w_0:y_0} q^1 \xrightarrow{w_1:y_1} \dots \text{ in } \mathcal{A}, \quad \text{and} \quad (\lambda^0_1, \dots, \lambda^0_{d+1}) \xrightarrow{y_0} (\lambda^1_1, \dots, \lambda^1_{d+1}) \xrightarrow{y_1} \dots \text{ in } U_{\text{parity}}.$$

Since vertices in U_{parity} satisfy the parity objective, $\liminf(y_0 y_1 \dots)$ is even, thus the above run in $\mathcal{A}$ is accepting, and so $w_0 w_1 \dots \in (q^0)^{-1}W$. ∎

Monotonicity Monotonicity of $U_{\mathcal{A}}$ follows from the structural assumptions over $\mathcal{A}$.

LEMMA 5.34. *The graph $U_{\mathcal{A}}$ is monotone.*

PROOF. Let

$$(q, \lambda_1, \dots, \lambda_{d+1}) \xrightarrow{a} (q', \lambda'_1, \dots, \lambda'_{d+1}) > (q'', \lambda''_1, \dots, \lambda''_{d+1}) \text{ in } U_{\mathcal{A}}.$$

We aim to prove that $(q, \lambda_1, \dots, \lambda_{d+1}) \xrightarrow{a} (q'', \lambda''_1, \dots, \lambda''_{d+1})$ in $U_{\mathcal{A}}$. By definition of the transitions of $U_{\mathcal{A}}$, there is a priority y such that $q \xrightarrow{a:y} q'$ in $\mathcal{A}$ and $(\lambda_0, \dots, \lambda_{d+1}) \xrightarrow{y} (\lambda'_0, \dots, \lambda'_{d+1})$ in U_{parity}.

We remark that, by definition of the order in $U_{\mathcal{A}}$, we have that $([q'']_0, \lambda''_1, \dots, \lambda''_{y-1}, [q'']_y) \leq ([q']_0, \lambda'_1, \dots, \lambda'_{y-1}, [q']_y)$ (for y even, similar if y odd). We distinguish four cases:

— If y is even and $([q']_0, \lambda'_1, \dots, \lambda'_{y-1}, [q']_y) = ([q'']_0, \lambda''_1, \dots, \lambda''_{y-1}, [q'']_y)$. Then in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:y+1} q''$ thus $q \xrightarrow{a:y} q''$, and in $U_{\mathcal{A}}$, $(\lambda_1, \dots, \lambda_{y-1}) \geq (\lambda'_1, \dots, \lambda'_{y-1}) = (\lambda''_1, \dots, \lambda''_{y-1})$, which concludes.

— If y is odd and $([q']_0, \lambda'_1, \dots, [q']_{y-1}, \lambda'_y) = ([q'']_0, \lambda''_1, \dots, [q'']_{y-1}, \lambda''_y)$. Then, in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:y} q''$ thus $q \xrightarrow{a:y} q''$, and in $U_{\mathcal{A}}$, $(\lambda_1, \dots, \lambda_y) > (\lambda'_1, \dots, \lambda'_y) = (\lambda''_1, \dots, \lambda''_y)$ which concludes.

— If for some even $x \leq y$ it holds that $([q']_0, \lambda'_1, [q']_2, \dots, \lambda'_{x-1}) = ([q'']_0, \lambda''_1, [q'']_2, \dots, \lambda''_{x-1})$ and $[q'']_x < [q']_x$. Then in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:x} q''$ thus $q \xrightarrow{a:x} q''$ and in $U_{\mathcal{A}}$, $(\lambda_1, \dots, \lambda_{x-1}) \geq (\lambda'_1, \dots, \lambda'_{x-1}) = (\lambda''_1, \dots, \lambda''_{x-1})$ thus $(\lambda_1, \dots, \lambda_{d+1}) \xrightarrow{x} (\lambda''_1, \dots, \lambda''_{d+1})$ which concludes.

— If for some even $x < y$ it holds that $([q']_0, \lambda'_1, \dots, [q']_x) = ([q'']_0, \lambda''_1, \dots, [q'']_x)$ and $\lambda'_{x+1} > \lambda''_{x+1}$. Then, in $\mathcal{A}$, $q \xrightarrow{a:y} q' \xrightarrow{\varepsilon:x+1} q''$ thus $q \xrightarrow{a:x+1} q''$ and in $U_{\mathcal{A}}$, $(\lambda_1, \dots, \lambda_{x+1}) \geq (\lambda'_1, \dots, \lambda'_{x+1}) > (\lambda''_1, \dots, \lambda''_{x+1})$ thus $(\lambda_1, \dots, \lambda_{d+1}) \xrightarrow{x+1} (\lambda''_1, \dots, \lambda''_{d+1})$ which concludes.

The other implication $v > v' \xrightarrow{a} v'' \implies v \xrightarrow{a} v''$ in $U_{\mathcal{A}}$ follows exactly the same lines. ■

Universality of $U_{\mathcal{A}}$ To prove Proposition 5.30, there remains to establish universality of $U_{\mathcal{A}}$, which follows easily from history-determinism of $\mathcal{A}$ and universality of U_{parity}.

LEMMA 5.35. *The graph $U_{\mathcal{A}}^{\top}$ is (κ, W)-universal.*

PROOF. We show universality for trees of $U_{\mathcal{A}}$ and conclude by Lemma 2.3. Let T be a Σ-tree of size $< \kappa$ that satisfies W. Let r be a sound resolver for $\mathcal{A}$. We define in a top-down fashion a labelling $\mathsf{r}_T : T \to Q$ such that, if $w_1 w_2 \dots w_k$ is the labelling of the path from the root to a vertex t, then $\mathsf{r}_T(t)$ is the target state of the run induced by r in $\mathcal{A}$. In particular, $t \xrightarrow{a} t'$ in T implies that $\mathsf{r}_T(t) \xrightarrow{a:x} \mathsf{r}_T(t')$ in $\mathcal{A}$ for some priority x, and, on each infinite branch $t_0 \xrightarrow{a_0} t_1 \xrightarrow{a_1} \dots$, the run $\mathsf{r}_T(t_0) \xrightarrow{a_0:x_0} \mathsf{r}_T(t_1) \xrightarrow{a_1:x_1} \dots$ is accepting in $\mathcal{A}$. Stated differently, the $[0, d+1]$-tree T_{parity} obtained from T by replacing each edge $t \xrightarrow{a} t'$ with the corresponding edge $t \xrightarrow{x} t'$ such that $\mathsf{r}_T(t) \xrightarrow{a:x} \mathsf{r}_T(t')$, satisfies the parity objective.

By (κ, parity)-universality of U_{parity}, there exists a morphism $\phi_{\text{parity}} : T_{\text{parity}} \to U_{\text{parity}}$. As T_{parity} has the same set of vertices than T, ϕ_{parity} defines a mapping from T to U_{parity}. We consider the product mapping $\phi = \mathsf{r}_T \times \phi_{\text{parity}} \colon T \to U_{\mathcal{A}}$ that sends $t \mapsto (\mathsf{r}_T(t), \phi_{\text{parity}}(t))$. It defines a morphism, as for any edge $t \xrightarrow{a} t'$ in T it holds that, for some x, $\mathsf{r}_T(t) \xrightarrow{a:x} \mathsf{r}_T(t')$ in $\mathcal{A}$ and $\phi_{\text{parity}}(t) \xrightarrow{x} \phi_{\text{parity}}(t')$ in $U_{\mathcal{A}}$. ■

This completes the proofs of the equivalence of the statements from Theorem 3.1 (excluding (3”)), providing a characterisation of positionality for ω-regular languages.

6. Two decision procedures

We now establish decidability of positionality of ω-regular languages in polynomial time, stated as Theorem 3.2 and prove the equivalence of Item (3”) with the other items in Theorem 3.1. We propose two decision procedures.

The first one follows our proof of Theorem 3.1 and attempts to build a deterministic signature automaton from a given deterministic parity automaton. We believe that the techniques used in such a procedure may prove interesting also in other contexts (see conclusion in Section 9.2).

The second procedure is simpler to describe: we use the fact that any (non-deterministic) automaton recognising a positional language is ε-completable (implication (5) $\implies$ (3”)). However, the proof itself relies on Theorem 3.4 about the closure under union (which only relies on the equivalence (1) $\iff$ (5) from Theorem 3.1).

6.1 Procedure 1: Recursive decomposition

The first decision procedure we present consists in, given a deterministic parity automaton $\mathcal{A}$, applying the construction from Section 5.2 to decide whether $W = \mathcal{L}(\mathcal{A})$ is positional. The general idea is simply to go through that construction, either ending up with a failure indicating that W is not positional, or with a deterministic structured signature automaton. If such an automaton is successfully obtained, it suffices to check full progress consistency, which can be done in polynomial time, as explained below.

Complexity of building a signature automaton Most proofs have been already given in Section 5.2; we also require the following easy lemma.

LEMMA 6.1. *Let $\mathcal{A}$ be a parity automaton and x a priority. Assume that $\mathcal{A}$ is deterministic over $\geq x$-transitions. Given two states q, p in $\mathcal{A}$, we can decide in polynomial time whether* $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)$.

We now detail the polynomial-time procedure. Let $\mathcal{A}$ be a deterministic parity automaton recognising W.

— First, we check for each pair of states q, p whether $\mathcal{L}(\mathcal{A}_q) \subseteq \mathcal{L}(\mathcal{A}_p)$, or $\mathcal{L}(\mathcal{A}_p) \subseteq \mathcal{L}(\mathcal{A}_q)$. If for some pair of states these languages are incomparable, then residuals of W are not totally ordered, and we can conclude that W is not positional.

Suppose that we have defined total preorders up to $\sqsubseteq_{x-2}$ making $\mathcal{A}$ a $(x-2)$-structured signature automaton.

— We $<x$-safe centralise $\mathcal{A}$, which can be done in polynomial time by Lemma 5.16. The obtained automaton is deterministic over $\geq x$-transitions.

— We compute the $<x$-safe components of $\mathcal{A}$, which can be done by doing a decomposition in SCCs of $\mathcal{A}|_{\geq x}$. We check whether, for each $<x$-safe component S and state q, the states in $S \cap [q]_{x-2}$ are totally preordered by inclusion of $<x$-safe languages, which can be done in polynomial time by Lemma 6.1. If this is not the case, we conclude that W is not positional.

— We remove the non-determinism from $\mathcal{A}$ – in polynomial time and without increasing the number of states – by applying Lemma 5.21.

— We compute (in polynomial time) the automaton $\mathcal{A}'$ given by Lemma 5.22 (see last subsection of Appendix A for details). We check whether $\mathcal{L}(\mathcal{A}') = W$, which can be done in polynomial time (testing equivalence of deterministic parity automata [23]). If this is not the case, we conclude that W is not positional.

After these operations, if we have not yet found that W is not positional, we obtain an equivalent deterministic automaton $\mathcal{A}'$ that is either x-structured signature, or strictly smaller than $\mathcal{A}$ (as given by Lemma 5.22).[11] In the former case, we continue defining preorders $\sqsubseteq_{x+1}$ and $\sqsubseteq_{x+2}$; in the latter, we restart from the beginning. In total, we repeat at most $d \cdot |Q|$ times a sequence of operations that take polynomial time.

Checking full progress consistency Assume that we have a deterministic structured signature automaton recognising $\mathcal{A}$. We cannot yet conclude that W is positional, as we do not know whether $\mathcal{A}$ is fully progress consistent, however, by Lemma 5.23, if W is positional this must be the case. By Theorem 3.1, this condition is also sufficient. We show now that we can check full progress consistency of $\mathcal{A}$ in polynomial time, finishing the proof of Theorem 3.2. For this, we generalise the method from [8, Lemma 25].

LEMMA 6.2. *Let $\mathcal{A}$ be a deterministic structured signature automaton. We can decide in polynomial time whether $\mathcal{A}$ is fully progress consistent.*

The proof crucially relies on the following lemma.

LEMMA 6.3. *A deterministic structured signature automaton $\mathcal{A}$ is fully progress consistent if and only if, for each even priority x and each pair of states q, p in $\mathcal{A}$ such that $q \sqsubset_x p$ we have:*

$$q \xrightarrow{w:\geq x} p \text{ and } p \xrightarrow{w:y} p \implies y \text{ is even.} \tag{1}$$

PROOF. It follows directly from the definition that a deterministic fully progress consistent automaton satisfies this property. To show the converse, assume that (1) holds; we aim to prove full progress consistency. Consider an even priority x and a word $w \in \Sigma^*$ such that $q \sqsubset_x p$ and $q \xrightarrow{w:\geq x} p$, we should prove $w^\omega \in \mathcal{L}(\mathcal{A}_q)$. We take x to be minimal such that $q \sqsubset_x p$, and thus we have $q \sim_{x-2} p$. For $x = 0$, the proof is identical to the one appearing in Lemma 4.6,

11 As mentioned before, we conjecture that the obtained automaton $\mathcal{A}'$ is always x-structured signature.

we assume that $x \geq 2$. Therefore, there is a $<x$-safe path connecting q and p, so we have $q \sim_{x-1} p$ (Item 2 from the definition of structured signature automaton), thus since $q \sqsubset_x p$ we get $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p)$ (Item 3). Consider the run over w^ω from q in $\mathcal{A}$:

$$\rho = q \xrightarrow{w:\geq x} p \xrightarrow{w:y_1} p_2 \xrightarrow{w:y_2} p_3 \xrightarrow{w:y_3} \cdots.$$

Since $w \in \mathsf{Safe}_{<x}(p)$ and $q \sqsubset_x p$, Lemma 5.11 yields $\mathsf{Safe}_{<x}(p) \subseteq \mathsf{Safe}_{<x}(p_2)$, and hence $y_1 \geq x$. Since $\sim_{x-2}$ is $[0, x-2]$-faithful and $q \xrightarrow{w:\geq x-2} p \sim_{x-2} q$, it follows that $p_2 \sim_{x-2} p$. Then it follows from $p \xrightarrow{w:\geq x} p_2$ that $p \sim_{x-1} p_2$, and thus $p \sqsubseteq_x p_2$. Applying the same reasoning by induction yields $y_i \geq x$ and $p \sqsubseteq_x p_i$ for all i, and thus $q \sqsubset_x p_i$

Eventually, ρ closes a cycle: there are N and k such that, for every $i \geq N$, $p_i = p_{i+k}$. We let $p' = p_{kN}$ and let y denote the minimal priority produced by the cycle. Then it holds that:

$$q \sqsubset_x p', \quad q \xrightarrow{w^{kN}:\geq x} p', \quad \text{and} \quad p' \xrightarrow{w^{kN}:y} p'.$$

Thus thanks to (1), y is even, and so $w^\omega = (w^{kN})^\omega \in \mathcal{L}(\mathcal{A}_q)$. ■

We can now deduce Lemma 6.2.

PROOF OF LEMMA 6.2. For each pair of states $q, p \in Q$ and each priority x, we define the languages of finite words

$$L^x_{q \to p} = \{w \in \Sigma^* \mid q \xrightarrow{w:x} p \text{ in } \mathcal{A}\}, \quad \text{and} \quad L^{\geq x}_{q \to p} = \{w \in \Sigma^* \mid q \xrightarrow{w:\geq x} p \text{ in } \mathcal{A}\}.$$

By Lemma 6.3, $\mathcal{A}$ is fully progress consistent if and only if, for each even priority $x \in [0, d]$ and each pair of states $q, p \in Q$ such that $q \sqsubset_x p$:

$$L^{\geq x}_{q \to p} \bigcap \Big(\bigcup_{y \text{ odd}} L^y_{p \to p} \Big) = \emptyset.$$

We show that for all pair of states, languages $L^{\geq x}_{q \to p}$ and $L^x_{q \to p}$ are regular and we can obtain deterministic finite automata for them in polynomial time. This implies that we can check the emptiness of intersections above in polynomial time, concluding the proof.

For $L^{\geq x}_{q \to p}$ the previous claim is clear: the finite automaton obtained by taking the automaton structure of $\mathcal{A}|_{\geq x}$ and taking q and p as initial and final states, respectively, is a finite automaton recognising $L^{\geq x}_{q \to p}$.

For $L^x_{q \to p}$, we consider the automaton over finite words that has as states $(Q \times [0, d]) \cup \{(q, \mathsf{init})\}$, and (q, init) and (p, x) as initial and final states, respectively. Transitions of the automaton are of the form $(q_1, x_1) \xrightarrow{a} (q_2, x_2)$ if the transition $q_1 \xrightarrow{a:y} q_2$ in $\mathcal{A}$ is such that $x_2 = \min\{x_1, y\}$. In words, this automaton keeps track of the run in $\mathcal{A}$ from q and of the minimal priority produced in the way. It accepts a word if it arrives to p while producing as minimal priority x, as we wanted. ■

6.2 Procedure 2: ε-completion

We now prove the following result.

THEOREM 6.4. *Let $\mathcal{A}$ be a non-deterministic parity automaton recognising a positional language W. Then for each pair of states $q, q' \in Q$, and for each even priority x, one may add (at least) one of the transitions*

$$q \xrightarrow{\varepsilon:x} q' \qquad \text{or} \qquad q' \xrightarrow{\varepsilon:x+1} q$$

without augmenting the language of $\mathcal{A}$.

Before proving Theorem 6.4, we argue that decidability of positionality in polynomial time (Theorem 3.2) follows. Let $\mathcal{A}_0$ be a deterministic parity automaton recognising a language W and using d as maximal priority (assumed even). We build an ε-completion of $\mathcal{A}_0$ as follows. At each step, pick a pair of states (q, q') such that neither $q \xrightarrow{\varepsilon:x} q'$ nor $q' \xrightarrow{\varepsilon:x+1} q$ belongs to the current automaton, for $x \leq d$ even. Then try to add one of these transitions, and see if the language increases (checking whether $\mathcal{L}(\mathcal{A}) \subseteq W$ can be done in polynomial time since W is recognised by the deterministic automaton $\mathcal{A}_0$ [23]). If the language does not increase for one of the two transitions, then proceed to the next step; otherwise conclude that W is not positional thanks to Theorem 6.4.

After $|Q|^2 d$ steps, we obtain an automaton such that for each pair of states (q, q') and for each even x, either $q \xrightarrow{\varepsilon:x} q'$ or $q' \xrightarrow{\varepsilon:x+1} q$. Now for each priority y, close the relations $\xrightarrow{\varepsilon:y}$ by transitivity, which does not augment the language. Moreover, for priorities $y \preccurlyeq y'$ (recall that $1 \preccurlyeq 3 \preccurlyeq \ldots \preccurlyeq d+1 \preccurlyeq d \preccurlyeq \ldots \preccurlyeq 2 \preccurlyeq 0$) add transition $q \xrightarrow{\varepsilon:y} q'$ whenever $\mathcal{A}$ contains $q \xrightarrow{\varepsilon:y'} q'$; this also does not augment the language. Then it holds that the relations $\xrightarrow{\varepsilon:1}, \xrightarrow{\varepsilon:3}, \ldots, \xrightarrow{\varepsilon:d+1}$ are total preorders refining one another, and that for each even x, $\xrightarrow{\varepsilon:x}$ is the strict variant of $\xrightarrow{\varepsilon:x+1}$. Stated differently, the obtained automaton $\mathcal{A}$ is an ε-completion of $\mathcal{A}_0$, which implies that W is positional thanks to (3) $\implies$ (5) in Theorem 3.1.

Note that on the way, we obtain the remaining implication (5) $\implies$ (3") from Theorem 3.1, stated as follows:

COROLLARY 6.5. *Any non-deterministic parity automaton recognising a positional language is ε-completable.*

We now prove Theorem 6.4. The proof is inspired by that of [24, Theorem 4.8], but it is more involved, because we now deal with parity automata rather than graphs (or safety automata). This difficulty is overcome thanks to Theorem 3.4.

PROOF OF THEOREM 6.4. Fix a pair of states $q, q' \in Q$ and an even priority x. Consider the game $\mathcal{G}$ defined as follows (see also Figure 18 below).

— The alphabet is $C = (\Sigma \cup \{\varepsilon\}) \times \{0, 1, \dots, d+1\} \times \{\mathsf{enter}_q, \mathsf{small}, \mathsf{neutral}\}$. Therefore, each edge has a letter in $\Sigma \cup \{\varepsilon\}$, a priority in $\{0, 1, \dots, d+1\}$, and a type in $\{\mathsf{enter}_q, \mathsf{small}, \mathsf{neutral}\}$. For a word $w \in C^\omega$, we write $w_\Sigma, w_{\mathsf{prio}}$ and w_{type} for the respective projections. For conciseness, we generally omit the type and write edges in the game as $\xrightarrow{a:y}$, just like in the automaton.

— The set of vertices consists in two copies of $\mathcal{A}$ indexed by q and q', together with an additional vertex $q_?$. Formally, $V = Q \times \{q, q'\} \cup \{q_?\}$. All vertices belong to Adam except for $q_?$ which belongs to Eve.

— The edges in the copy indexed by q (resp. q') follow exactly the transitions in $\mathcal{A}$, except those leading to q' (resp. q), which are instead redirected to $q_?$ (but keep the same letter and priority).

— The vertex $q_?$ has exactly two outgoing edges: $q_? \xrightarrow{\varepsilon:x+1} (q, q)$ and $q_? \xrightarrow{\varepsilon:x} (q', q')$.

— The edge $q_? \xrightarrow{\varepsilon:x+1} (q, q)$ has type enter_q, and edges with priority $\leq x$ inside the copy indexed by q have type small. All other edges are neutral.

— The objective is

$$W_{\mathcal{G}} = W_\Sigma \cup \mathsf{oddParity} \cup \mathsf{goodType},$$

where W_Σ is the set of words w such that $w_\Sigma \in W$, oddParity is the set of words whose minimal priority appearing infinitely often is odd, and goodType is the set of words with infinitely many occurrences of type enter_q and finitely many occurrences of type small.

Clearly W_Σ is positional and so is oddParity; likewise, goodType rewrites as a parity condition (by mapping small to 1, enter_q to 2 and neutral to 3), and thus it is also positional. Moreover, oddParity and goodType are prefix-independent. It follows from Theorem 3.4 that $W_{\mathcal{G}}$ is positional.

CLAIM 6.6. *Eve wins from (q_{init}, q) and from (q_{init}, q') in $\mathcal{G}$.*

Proof. Consider the strategy which, whenever reaching $q_?$ from an edge of the form $(p, q') \to q_?$, reads the edge $q_? \xrightarrow{\varepsilon:x+1} (q, q)$ and whenever reaching $q_?$ from an edge of the form $(p, q) \xrightarrow{q} {}_?$, reads the edge $q_? \xrightarrow{\varepsilon:x} (q', q')$. Take an infinite path $\pi_{\mathcal{G}}$ in $\mathcal{G}$ from (q_{init}, q) which is consistent with the above strategy. It is of the form

$$\pi_{\mathcal{G}} : (q_{\mathsf{init}}, q) \xrightarrow{w_0:y_0} (p_0, q) \xrightarrow{a_0:z_0} q_? \xrightarrow{\varepsilon:x} (q', q') \xrightarrow{w_1:y_1} (p_1, q') \xrightarrow{a_1:z_1} q_? \xrightarrow{\varepsilon:x+1} (q, q) \xrightarrow{w_2:y_2} \dots,$$

where $w_0, p_0, a_0, w_1, p_1, a_1, \dots$ is either infinite (and the w_i's are finite) or it ends with some w_i which is infinite. We aim to prove that $\pi_{\mathcal{G}}$ is winning, that is, its label belongs to $W_{\mathcal{G}}$.

Observe that

$$\pi_{\mathcal{A}} : q_{\mathsf{init}} \xrightarrow{w_0:y_0} p_0 \xrightarrow{a_0:z_0} q' \xrightarrow{w_1:y_1} p_1 \xrightarrow{a_1:z_1} q \xrightarrow{w_2:y_2} \dots$$

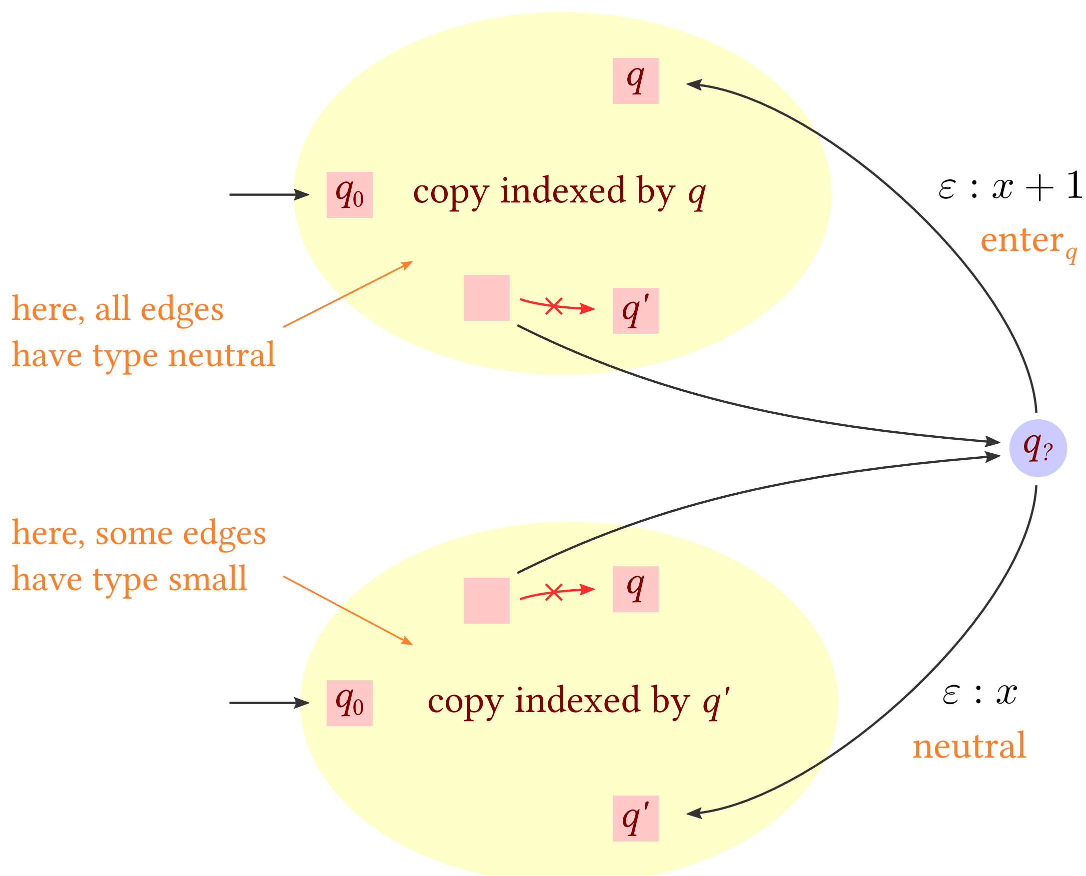

Figure 18. The game $\mathcal{G}$ in the proof of Theorem 6.4. Edges which are crossed out are those that are redirected (while keeping the same label). Note that (q', q) and (q, q') are not reachable; these vertices could be removed from the game.

defines a path in $\mathcal{A}$. If $\pi_{\mathcal{A}}$ satisfies W, then $\pi_{\mathcal{G}}$ satisfies W_Σ (since we only add some occurrences of ε to the projections on $\Sigma \cup \{\varepsilon\}$). Thus we assume that $\pi_{\mathcal{A}}$ does not satisfy W, which means that the run in the automaton is rejecting. If $\pi_{\mathcal{G}}$ satisfies goodType then it is winning, so we assume otherwise: there are either finitely many occurrences of enter$_q$ or infinitely many occurrences of small.

- If there are finitely many occurrences of enter$_q$ in $\pi_{\mathcal{G}}$, then the priorities in $\pi_{\mathcal{G}}$ eventually coincide with those in $\pi_{\mathcal{A}}$, and since this is a rejecting run in $\mathcal{A}$, $\pi_{\mathcal{G}}$ satisfies oddParity.
- If there are infinitely many occurrences of small, then the minimal priority appearing infinitely often in $\pi_{\mathcal{A}}$ is $\leq x$, therefore adding (even infinitely many) priorities x and $x+1$ does not change the fact that the run is rejecting. Therefore $\pi_{\mathcal{G}}$ satisfies oddParity. ◆

Since Eve wins and $W_{\mathcal{G}}$ is positional, she has a winning positional strategy σ.

CLAIM 6.7. *If $\sigma(q_?) = q_? \xrightarrow{\varepsilon:x} q'$, then adding the transition $q \xrightarrow{\varepsilon:x} q'$ to $\mathcal{A}$ does not augment the language.*

Proof. Let $\mathcal{A}'$ be the automaton obtained by adding to $\mathcal{A}$ the transition $q \xrightarrow{\varepsilon:x} q'$. Consider an accepting run $\pi_{\mathcal{A}'}$ from q_{init} in $\mathcal{A}'$. Decompose it around the occurrences of $q \xrightarrow{\varepsilon:x} q'$ as follows:

$$\pi_{\mathcal{A}'} : q_{\mathsf{init}} \overset{w_0:y_0}{\rightsquigarrow} p_0 \xrightarrow{a_0:z_0} q \xrightarrow{\varepsilon:x} q' \overset{w_1:y_1}{\rightsquigarrow} p_1 \xrightarrow{a_1:z_1} q \xrightarrow{\varepsilon:x} q' \overset{w_2:y_2}{\rightsquigarrow} \dots,$$

where the sequence $w_0, p_0, a_0, w_1, p_1, a_1, \dots$ is either infinite (and the w_i are finite), or it is finite and ends with some w_i which is infinite.

Then

$$\pi_{\mathcal{G}} : (q_{\mathsf{init}}, q') \overset{w_0:y_0}{\rightsquigarrow} (p_0, q') \xrightarrow{a_0:z_0} q_? \xrightarrow{\varepsilon:x} (q', q') \overset{w_1:y_1}{\rightsquigarrow} p_1 \xrightarrow{a_1:z_1} q_? \xrightarrow{\varepsilon:x} (q', q') \overset{w_2:y_2}{\rightsquigarrow} \dots$$

defines a path in $\mathcal{G}$ which is consistent with σ. Therefore the label of $\pi_{\mathcal{G}}$ satisfies $W_{\mathcal{G}}$. Note that priorities in $\pi_{\mathcal{A}'}$ and $\pi_{\mathcal{G}}$ are the same, and $\pi_{\mathcal{A}'}$ is accepting, therefore $\pi_{\mathcal{G}}$ does not satisfy oddParity. Moreover, $\pi_{\mathcal{G}}$ has no occurrence of enter_q, thus it does not satisfy goodType. We conclude that $\pi_{\mathcal{G}}$ satisfies W_Σ, and thus $w_0 a_0 w_1 a_1 \dots \in W$, so $\mathcal{L}(\mathcal{A}') \subseteq W$. ◆

The proof of the other case is similar to the previous one, with a slight difference in the analysis.

CLAIM 6.8. *If $\sigma(q_?) = q_? \xrightarrow{\varepsilon:x+1} q$, then adding the transition $q' \xrightarrow{\varepsilon:x+1} q$ to $\mathcal{A}$ does not augment the language.*

Proof. Let $\mathcal{A}'$ be the automaton obtained by adding to $\mathcal{A}$ the transition $q' \xrightarrow{\varepsilon:x+1} q$. Consider an accepting run $\pi_{\mathcal{A}'}$ from q_{init} in $\mathcal{A}'$. Decompose it around the occurrences of $q' \xrightarrow{\varepsilon:x+1} q$ as follows:

$$\pi_{\mathcal{A}'} : q_{\mathsf{init}} \overset{w_0:y_0}{\rightsquigarrow} p_0 \xrightarrow{a_0:z_0} q' \xrightarrow{\varepsilon:x+1} q \overset{w_1:y_1}{\rightsquigarrow} p_1 \xrightarrow{a_1:z_1} q' \xrightarrow{\varepsilon:x+1} q \overset{w_2:y_2}{\rightsquigarrow} \dots,$$

where the sequence $w_0, p_0, a_0, w_1, p_1, a_1, \dots$ is either infinite (and the w_i are finite), or it is finite and ends with some w_i which is infinite.

Then

$$\pi_{\mathcal{G}} : (q_{\mathsf{init}}, q) \overset{w_0:y_0}{\rightsquigarrow} (p_0, q) \xrightarrow{a_0:z_0} q_? \xrightarrow{\varepsilon:x+1} (q, q) \overset{w_1:y_1}{\rightsquigarrow} p_1 \xrightarrow{a_1:z_1} q_? \xrightarrow{\varepsilon:x+1} (q, q) \overset{w_2,y_2}{\rightsquigarrow} \dots$$

defines a path in $\mathcal{G}$ which is consistent with σ. Therefore the label of $\pi_{\mathcal{G}}$ satisfies $W_{\mathcal{G}}$. Note that priorities in $\pi_{\mathcal{A}'}$ and $\pi_{\mathcal{G}}$ are the same, and $\pi_{\mathcal{A}'}$ is accepting, therefore $\pi_{\mathcal{G}}$ does not satisfy oddParity.

Now note that $\pi_{\mathcal{A}'}$ has infinitely many occurrences of priority $x+1$ and yet it is accepting, therefore it has infinitely many occurrences of priorities $\leq x$. Thus $\pi_{\mathcal{G}}$ has infinitely many occurrences of small, so it does not satisfy goodType. We conclude that $\pi_{\mathcal{G}}$ satisfies W_Σ, and therefore the label $w_0 a_0 w_1 a_1 \dots$ of $\pi_{\mathcal{A}'}$ belongs to W. ◆

This concludes the proof. ■

REMARK 6.9. It is interesting to remark that our proof relies on the fact that for any ω-regular positional objective W, the objective $W \cup$ parity is positional. We do not know a direct proof of this fact (without using Theorem 3.4 which relies on the machinery employed to prove Theorem 3.1). Such a direct proof would give, together with Theorem 6.4, an easier path to the main characterisation and the polynomial time decidability (though it would fall short of establishing the 1-to-2-player lift and the finite-to-infinite lift).

7. Bipositionality of all objectives

In this section we provide a characterisation of all bipositional objectives, without ω-regularity or prefix-independence assumptions. This characterisation extends the result of Colcombet and Niwiński [27], who showed that the only prefix-independent bipositional objective (over all game graphs) is the parity objective. Recently, Bouyer, Randour and Vandenhove [11] generalised that result in an orthogonal direction: they proved that the only objectives for which both players can play optimally using finite chromatic memory are ω-regular objectives.

7.1 Characterisation of bipositionality and consequences

We say that an objective $W \subseteq \Sigma^\omega$ is *bi-progress consistent* if both W and its complement are progress consistent, that is, if it satisfies that for all residual class $[u]$ and finite word $w \in \Sigma^*$:

- $[u] < [uw] \implies uw^\omega \in W$, and
- $[uw] < [u] \implies uw^\omega \notin W$.

THEOREM 7.1 (Characterisation of bipositionality). *An objective $W \subseteq \Sigma^\omega$ is bipositional (over all games) if and only if:*

1. *W has a finite number of residuals, totally ordered by inclusion, and*
2. *W is bi-progress consistent, and*
3. *W can be recognised by a parity automaton on top of the automaton of residuals.*

This characterisation only holds for infinite games, as there are non ω-regular objectives that are bipositional over finite games, as, for example, energy objectives [9] and their generalisation [37]. However, we deduce from Theorem 3.3 that in the case of ω-regular objectives these conditions do also characterise bipositionality over finite games.

COROLLARY 7.2 (Bipositionality over finite games for ω-regular objectives). *An ω-regular objective $W \subseteq \Sigma^\omega$ is bipositional over finite games if and only if it satisfies the three conditions from Theorem 7.1.*

Consequences: Lifts and decidability For ω-regular objectives, we can directly lift the corollaries of Theorem 3.1 obtained for positionality to bipositionality. For non-ω-regular ones, the finite-to-infinite lift does not hold, as commented above. On the other hand, a combination with a recent result from Bouyer, Randour and Vandenhove [11, Theorem 3.8] implies that the 1-to-2-player lift holds for any objective.

We say that an objective $W \subseteq \Sigma^\omega$ is *bipositional over (finite) Eve and Adam-games* if both W and its complement $\Sigma^\omega \setminus W$ are positional over (finite) Eve-games.

COROLLARY 7.3 (1-to-2 player lift of bipositionality). *An objective $W \subseteq \Sigma^\omega$ is bipositional (over all games) if and only if it is bipositional over Eve and Adam-games.*

We note that a 1-to-2 player lift was obtained for objectives that are bipositional over finite game graphs by Gimbert and Zielonka [32, 57] (even in the more general setting of qualitative objectives). However, their proof consisted in an induction over the size of the game graph, so it does not generalise to infinite games. Indeed, as remarked above, bipositionality over finite and infinite graphs behaves in a completely different manner. In this respect, Corollary 7.3 and the result of Gimbert and Zielonka are incomparable.

COROLLARY 7.4 (Finite-to-infinite lift of bipositionality for ω-regular objectives). *An ω-regular objective $W \subseteq \Sigma^\omega$ is bipositional (over all games) if and only if it is bipositional over finite Eve and Adam-games.*

We obtain decidability for bipositionality in polynomial time from its counterpart in the case of positionality (Theorem 3.2). We observe that Theorem 7.1 provides an alternative way to check bipositionality.

COROLLARY 7.5 (Decidability of bipositionality). *Given a deterministic parity automaton $\mathcal{A}$, we can decide in polynomial time whether $\mathcal{L}(\mathcal{A})$ is bipositional.*

An example

EXAMPLE 7.6 (Parity over occurrences). We let $\Sigma = [0, d]$ and let $W_{\mathsf{OccParity}}$ be the language of words such that the minimal priority appearing on them is even:

$$W_{\mathsf{OccParity}} = \{w \in [0, d]^\omega \mid \min(w) \text{ is even}\}.$$

An automaton recognising W is depicted in Figure 19. It has one state per residual, which are totally ordered, and it is immediate to check that it is bi-progress consistent. Therefore, $W_{\mathsf{OccParity}}$ is a bipositional objective. ◆

Some more complex examples can be generated by, for example, adding some output priorities to the automaton above without breaking the bi-progress consistency condition. However, the combination of being recognisable by the automaton of residuals with bi-progress consistency greatly restricts the possibilities of generating examples of bipositional languages. We wonder whether a more precise characterisation of languages satisfying these three properties can be obtained.

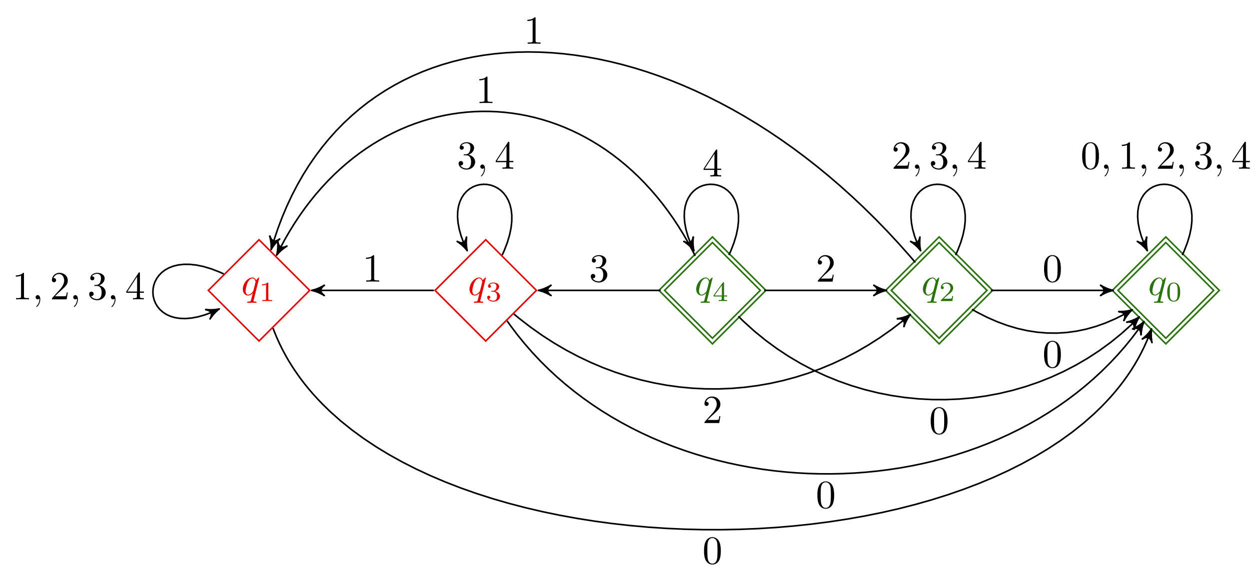

Figure 19. Automaton recognising $W_{\text{OccParity}}$, for $d = 4$. The initial state is q_4. This automaton is in fact what is sometimes called a weak automaton: runs that finally end in an even state are accepting, and those ending in an odd state are rejecting.

7.2 Proof of the characterisation

Necessity of the conditions The necessity of the total order over the residuals of W is given by Lemma 4.1, and the necessity of bi-progress consistency is given by Lemma 4.6. The following lemma provides the necessity of the last condition of Theorem 7.1. It can be obtained by instantiating the first item of [11, Theorem 3.6] for the case of bipositional objectives.

LEMMA 7.7 ([11, Theorem 3.6]). *If $W \subseteq \Sigma^\omega$ is bipositional over Eve and Adam-games, then W is ω-regular and can be recognised by a parity automaton on top of the automaton of residuals.*

Sufficiency of the conditions The sufficiency of conditions of Theorem 7.1 can be shown by providing well-ordered monotone universal graphs for W and its complement. An even simpler option – now that we have already done such construction for positional objectives – is to provide fully progress consistent signature automata recognising these languages, and then use the characterisation of positionality given by Item (2) from Theorem 3.1.

We show that the parity automaton on top of the automaton of residuals is a fully progress consistent signature automaton. Since by hypothesis $\mathsf{Res}(W)$ is totally ordered, the order $\sqsubseteq_0$ given by inclusion of residuals satisfies the first requirement of the definition of signature automaton. As the residual classes of this automaton are singletons, we can define all relations $\sim_x$ to be the equality relation too, so this automaton satisfies all the requirements to be a signature automaton. Moreover, as W is progress consistent and the $\sim_x$-classes of the automaton are singletons, it is also fully progress consistent. The argument is symmetric for $\Sigma^\omega \setminus W$.

8. Positionality of closed and open objectives

8.1 Closed objectives

We recall that an objective W is closed if

$$W = \mathsf{Safety}(L) = \{w \mid w \text{ does not contain any prefix in } L\},$$

for some language of finite words L. Positional closed objectives were first characterised by Colcombet, Fijalkow and Horn [25], as those that have a totally ordered set of residuals. However, as they already remarked, this characterisation only holds for finite branching game graphs.

We now give a characterisation of positionality over all game graphs for closed objectives. Namely, a closed objective W is positional if and only if $\mathsf{Res}(W)$ is well-ordered by inclusion (Theorem 8.4) (in fact, the well-foundedness of $\mathsf{Res}(W)$ is a necessary condition for finite memory determinacy of any objective.) This (and its generalisation to memory) was already observed in [20].

8.1.1 Well-foundedness of residuals

Next example, taken from [25], shows that total order over residuals does not suffice to ensure positionality of arbitrary closed objectives.

EXAMPLE 8.1 (Outbidding game [25]: Total order does not suffice). Let $\Sigma = \{a, b, c\}$ and L be the language of finite words:

$$L = \{w \in \Sigma^* \mid \text{ for some } u \in \Sigma^* \text{ with } |u|_a \leq |u|_b, uc \text{ is a prefix of } w\},$$

where $|u|_x$ is the number of occurrences of a in u. We consider the closed objective $W = \mathsf{Safety}(L)$. The residuals of W are totally ordered by inclusion:

$$\emptyset = c^{-1}W < \cdots < (a^n)^{-1}W < \cdots < a^{-1}W < \varepsilon^{-1}W < b^{-1}W < \cdots .$$

However, W is not positional, as witnessed by the game in Figure 20. ◆

LEMMA 8.2 (Necessity of the well-order of residuals). *Let $W \subseteq \Sigma^\omega$ be an objective that is positional over ε-free Eve-games. Then,* $\mathsf{Res}(W)$ *is well-ordered by inclusion.*

We have already seen (Lemma 4.1), that if W is positional, then $\mathsf{Res}(W)$ is totally ordered. We need to prove that $\mathsf{Res}(W)$ is well-founded.

LEMMA 8.3 (Well-foundedness of residuals necessary for finite memory). *Let $W \subseteq \Sigma^\omega$ be an objective such that Eve can play optimally using positional strategies (or using finite memory strategies) over ε-free Eve-games. Then,* $\mathsf{Res}(W)$ *is well-founded (for the order given by inclusion of residuals).*

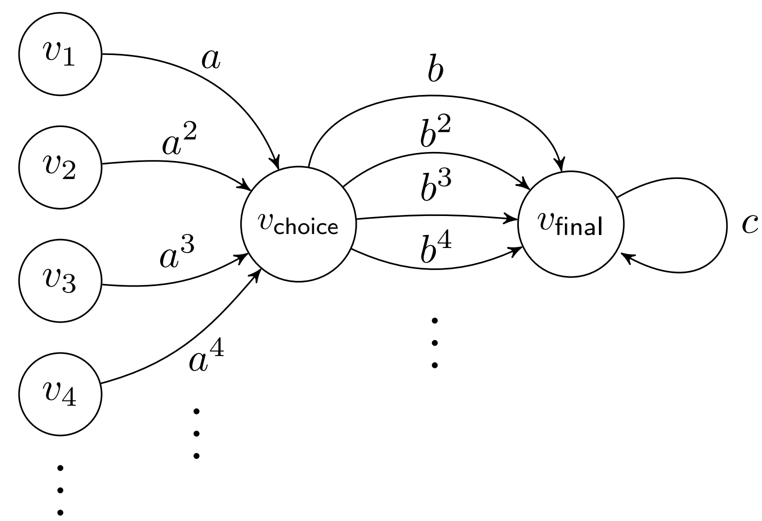

Figure 20. Outbidding game from Example 8.1. First, a sequence a^n is produced, for some $n \in \mathbb{N}$. In order to win, Eve needs to answer with b^m, with $m > n$. Therefore, she can win from any vertex in the game, but no positional strategy guarantees the victory from all states.

PROOF. Suppose by contradiction that there is an infinite strictly decreasing sequence of residuals:

$$u_1{}^{-1}W \supsetneq u_2{}^{-1}W \supsetneq \cdots,\ u_i \in \Sigma^*.$$

(We suppose without loss of generality that $\varepsilon \neq u_i$ for all i.) Let $w_i \in \Sigma^\omega$ such that $w_i \in u_i{}^{-1}W \setminus u_{i+1}{}^{-1}W$. We consider the game – similar to the outbidding game from Figure 20 – in which a word u_i labels a path from a vertex v_i to v_{choice}, for each i. From this latter vertex, Eve can choose a between paths labelled by $\{w_i \mid i \in \mathbb{N}\}$. Eve can win be answering w_i to u_i. However, any positional (or finite memory) strategy will only consider a finite number of responses $w_{j_1}, \dots, w_{j_n}$. Therefore, such a strategy is losing from v_K for $K > \max\{j_t \mid 1 \leq t \leq n\}$. ■

8.1.2 Characterisation for closed objectives

THEOREM 8.4 (Positional closed objectives). *Let $W \subseteq \Sigma^\omega$ be a closed objective. Then, W is positional (over all game graphs) if and only if* $\mathsf{Res}(W)$ *is well-ordered by inclusion.*

PROOF. We have already shown that this condition is necessary. To prove sufficiency, we give, for each cardinal κ, a (κ, W)-universal well-ordered monotone graph. We conclude by Proposition 2.2.

Let U be the Σ-graph that has as vertices $\mathsf{Res}(W) \setminus \{\emptyset\}$, ordered by inclusion. By hypothesis, this is a well-order. For each $a \in \Sigma$, we let

$$u^{-1}W \xrightarrow{a} u'^{-1}W \quad \text{iff} \quad u'^{-1}W \leq (ua)^{-1}W.$$

We note that if ua is already losing ($(ua)^{-1}W = \emptyset$), then transition $u^{-1}W \xrightarrow{a} u'^{-1}W$ does not appear in U. By Lemma 2.17, graph U is monotone. The hypothesis of closeness of W is fundamentally used in next claim.

CLAIM 8.5. *A vertex $u^{-1}W$ of U satisfies W if and only if $u^{-1}W \subseteq \varepsilon^{-1}W = W$.*

Proof. Let $L \subseteq \Sigma^*$ such that $W = \mathsf{Safety}(L)$. Let $u_1{}^{-1}W \xrightarrow{a_1} u_2{}^{-1}W \xrightarrow{a_2} \dots$ be a path in U from $u_1 = u$. By induction we obtain $\emptyset \neq u_i{}^{-1}W \subseteq (u_1a_1 \dots a_{i-1})^{-1}W$. Therefore, for all i, $ua_1 \dots a_i \notin L$, so, by definition of W, the infinite word $ua_1a_2 \dots$ belongs to W. ◆

We show that U is (κ, W)-universal for trees, for every cardinal κ, and conclude by Lemma 2.3. Let T be a Σ-tree which satisfies W. For each node $t \in T$, let $\phi(t) = u_t{}^{-1}W$ be the minimal residual such that t satisfies $u_t{}^{-1}W$. In particular, for the root t_0, $\phi(t_0)$ satisfies W by the previous claim. We claim that ϕ is a morphism. Indeed, if $t \xrightarrow{a} t'$ in T and t satisfies $u^{-1}W$, then t' satisfies $(ua)^{-1}W$. Therefore, $u_t'^{-1}W \leq (u_ta)^{-1}W$, so $\phi(t) = u_t{}^{-1}W \xrightarrow{a} u_{t'}{}^{-1}W = \phi(t')$ is an edge in U. ■

8.2 Open objectives

We recall that an objective W is open if

$$W = \mathsf{Reach}(L) = \{w \mid w \text{ contains some prefix in } L\},$$

for some $L \subseteq \Sigma^*$.

8.2.1 Reset-stability

In Section 4.2, we showed that positional ω-regular open objectives are exactly those with residuals totally ordered and that are progress consistent. However, for non-ω-regular objectives, these conditions do not suffice, even if residuals are well-ordered.

EXAMPLE 8.6 (Progress consistency does not suffice). Let $\Sigma = \mathbb{N}$ and let W be the set of sequences that are not strictly-increasing:

$$W = \{a_1a_2 \dots \in \mathbb{N}^\omega \mid a_{i+1} \leq a_i \text{ for some } i\}.$$

This objective is open, as $W = \mathsf{Reach}(\text{two consecutive non-increasing numbers})$. Its residuals are:

$$\varepsilon^{-1}W < 0^{-1}W < 1^{-1}W < 2^{-1}W < \dots < (00)^{-1}W = \Sigma^\omega.$$

Therefore, $\mathsf{Res}(W)$ is well-ordered. Moreover, W is progress consistent: any repetition of factors induces a non-strict inequality < between consecutive letters.

However, we claim that W is not positional. Consider the game in Figure 21. Eve can win this game: no matter what is the vertex v_i chosen by Adam, she can first move one position

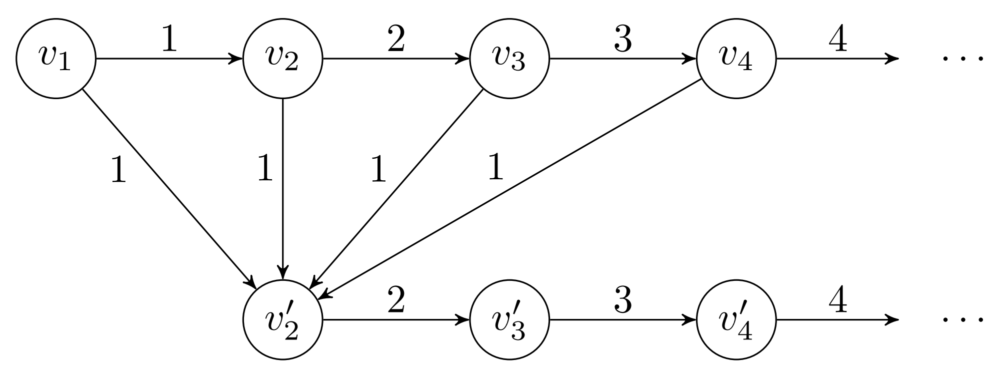

Figure 21. Game in which Eve wins if she does not produce a strictly-increasing sequence of numbers. She can win from every vertex, but not positionally.

to the right, producing i, and then go down producing letter 1. This ensures two consecutive non-increasing numbers. However, she cannot win positionally. Indeed, if such a strategy tells her to go always to the right, the sequence produced will be strictly increasing. If she choses to go down in vertex v_i, Adam can win by initialising the play in that vertex. ◆

DEFINITION 8.7 (Reset-stability). We say that an objective $W \subseteq \Sigma^\omega$ is *reset-stable* if, for each sequence of finite words $u_1, u_2, u_3, \dots \in \Sigma^+$ and of residuals $s_0{}^{-1}W, s_1{}^{-1}W, s_2{}^{-1}W, \dots \in \mathsf{Res}(W)$:

$$s_i{}^{-1}W \subsetneq (s_{i-1}u_i)^{-1}W \ \text{ for all } i \geq 1 \quad \Longrightarrow \quad u_1u_2u_3\cdots \in s_0{}^{-1}W.$$

An intuitive idea of reset-stability is the following. Consider the (potentially infinite) automaton of residuals of W, which inherits the order over residuals. Add to it all ε-transitions going backwards: $s^{-1}W \xrightarrow{\varepsilon} s'^{-1}W$ for $s'^{-1}W < s^{-1}W$. When a run takes an ε-transition, we say that it *makes a reset*. What reset-stability tells us is that any run making infinitely many resets must be accepting. The words $u_1, u_2, \dots$ in the definition above correspond to fragments where no reset takes place, and $s_i{}^{-1}W$ is the residual where we land after the i$^{\text{th}}$ reset. (See also the notion of 0-jumps in the proof of Lemma 5.28).

REMARK 8.8. If W is reset-stable, it is progress consistent. The converse holds if $\mathsf{Res}(W)$ is finite and totally ordered.

We note that all closed objectives are reset-stable.

LEMMA 8.9 (Necessity of reset-stability). *Let $W \subseteq \Sigma^\omega$ be a positional objective over ε-free Eve-games. Then, W is reset-stable.*

PROOF. Suppose by contradiction that W is not reset-stable. That is, there are $u_1, u_2, \dots \Sigma^+$ and $s_0{}^{-1}W, s_1{}^{-1}W \dots$ such that $s_i{}^{-1}W < (s_{i-1}u_i)^{-1}W$, but $u_1u_2\cdots \notin s_0{}^{-1}W$.

Let $w_i \in \Sigma^\omega$ such that $w_i \in (s_{i-1}u_i)^{-1}W \setminus s_i{}^{-1}W$. We consider the game pictured in Figure 22 (to ensure it to be ε-free, we just remove vertex v_i if $s_i = \varepsilon$).

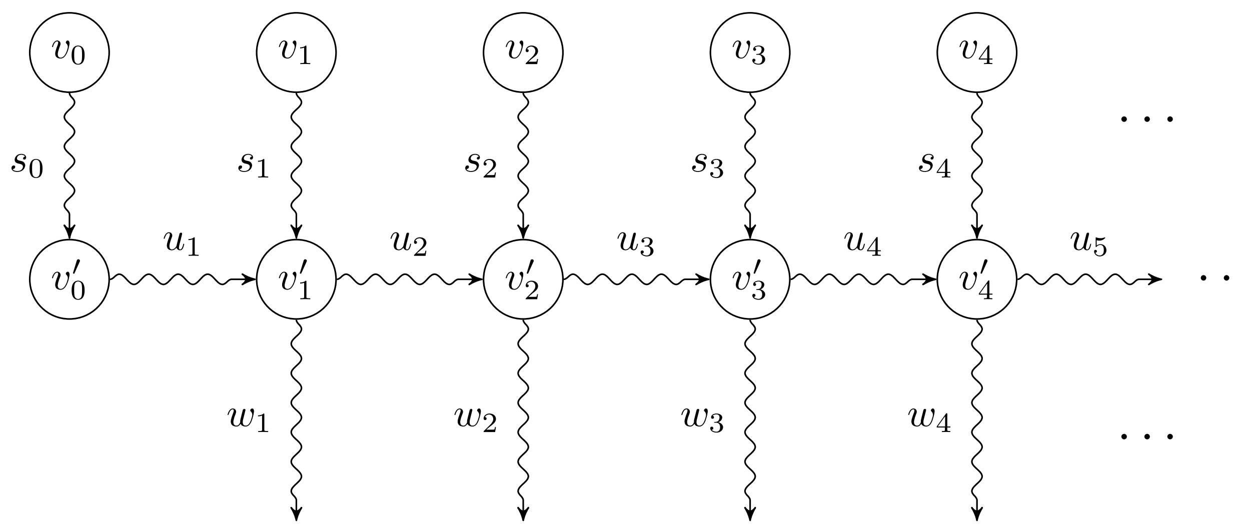

Figure 22. Game in the proof of Lemma 8.9. Eve can win from every v_i, but not positionally.

Eve can win $\mathcal{G}$ from any vertex v_i (and from v'_i if $s_i = \varepsilon$), as she can produce the word $s_i u_{i+1} w_{i+1}$, which belongs to W, as we have taken $w_{i+1} \in (s_i u_{i+1})^{-1} W$. However, we show that no positional strategy ensures to win from all these vertices. We distinguish two cases. If this strategy takes the path $v'_i \xrightarrow{u_{i+1}} v'_{i+1}$ for all i, then it is not winning from v_0, as by hypothesis $u_1 u_2 \cdots \notin s_0{}^{-1} W$. If on the contrary this strategy takes a path $v'_i \xrightarrow{w_i}$, then it is not winning from v_i. ∎

8.2.2 Characterisation for open objectives

THEOREM 8.10 (Positional open objectives). *Let $W \subseteq \Sigma^\omega$ be an open objective. Then, W is positional (over all game graphs) if and only if:*

— $\mathsf{Res}(W)$ *is well-ordered by inclusion, and*

— *W is reset-stable.*

PROOF. The necessity of the conditions has already been established in Lemmas 8.3 and 8.9. To prove the sufficiency, we give, for each cardinal κ a well-ordered monotone graph that is (κ, W)-universal for trees, and conclude by Proposition 2.2 and Lemma 2.3.

We let U be the Σ-graph having as set of vertices $(\mathsf{Res}(W) \setminus \{\emptyset\}) \times \kappa$, ordered lexicographically. This graph is well-ordered, as by hypothesis so is $\mathsf{Res}(W)$. The edges are given by:

$$(u^{-1}W, \lambda) \xrightarrow{a} (u'^{-1}W, \lambda') \quad \text{if} \quad \begin{cases} u'^{-1}W = (ua)^{-1}W \text{ and } \lambda' < \lambda, & \text{or} \\ u'^{-1}W \subsetneq (ua)^{-1}W, & \text{or} \\ u^{-1}W = \Sigma^\omega. \end{cases}$$

By Lemma 2.17, this graph is monotone. We show its (κ, W)-universality for trees. The next claim, which relies in the reset-stability hypothesis, provides the key ingredient for this. We let L be the language of finite words such that $W = \mathsf{Reach}(L)$.

CLAIM 8.11. *For each ordinal $\lambda < \kappa$ and each residual $u^{-1}W$, the vertex $(u^{-1}W, \lambda)$ satisfies $u^{-1}W$ in U (i.e. for all paths from $(u^{-1}W, \lambda)$ with label w, it holds that $uw \in W$).*

Proof. Let $\rho = (u_0{}^{-1}W, \lambda_0) \xrightarrow{a_0} (u_1{}^{-1}W, \lambda_1) \xrightarrow{a_1} \cdots$ be an infinite path from $(u^{-1}W, \lambda)$ in U. If $u_{i+1}{}^{-1}W \subsetneq (u_i a_i)^{-1}W$, we say that the transition $\xrightarrow{a_i}$ makes a reset. By induction, we obtain that $u_i{}^{-1}W \subseteq (ua_1 \dots a_{i-1})^{-1}W$. We distinguish two cases: (1) If ρ makes infinitely many resets, then we conclude by reset-stability. (2) If ρ makes finitely many resets, then eventually $\lambda_{i+1} < \lambda_i$ for all i, unless $u_i{}^{-1}W = \Sigma^\omega$. We conclude that eventually $u_i{}^{-1}W = \Sigma^\omega \subseteq (ua_1 \dots a_{i-1})^{-1}W$. Therefore, $ua_1 \dots a_{i-1} \in L$, so $ua_1a_2 \cdots \in W$. ◆

Let T be a Σ-tree whose root satisfies W. We give a morphism $\phi\colon T \to U$, which we decompose in $\phi_1\colon T \to \mathsf{Res}(W) \setminus \{\emptyset\}$ and $\phi_2\colon T \to \kappa$. For each $t \in T$, let u_t be the word labelling the path from the root t_0 to t. We let $\phi_1(t) = u_t{}^{-1}W$ for each t. We define ϕ_2 by transfinite induction. By hypothesis, each branch eventually contains vertices t such that $u_t \in L$ (that is, $u_t^{-1} = \Sigma^\omega$). For all these vertices, we let $\phi_2(t) = 0$. The tree obtained by removing these vertices, named T_1, does not have any infinite branch. For an ordinal $\lambda < \kappa$, let T_λ be the set of nodes for which we have not defined ϕ_2 at step λ of the induction. For each leaf t of T_λ, we let $\phi_2(t) = \lambda$.

This mapping has the two following properties:

— if $t \xrightarrow{a} t'$ in T, then $\phi_1(t') = (u_t a)^{-1}W$, and

— if $t \xrightarrow{a} t'$ in T, either $\phi_2(t') < \phi_2(t)$, or $u_{t'} \in L$.

This ensures that $\phi = (\phi_1, \phi_2)$ is a morphism, concluding the proof. ■

EXAMPLE 8.12 (Positional open objective). Let $\Sigma = \mathbb{N}$ and let W be the set of sequences that start by $123 \dots n$, and eventually decrease. Formally:

$$W = \{a_1a_2 \cdots \in \mathbb{N}^\omega \mid \text{there is } j \text{ such that } a_i = i \text{ for } i < j \text{ and } a_j < j\}.$$

This objective is recognised by the infinite reachability automaton depicted in Figure 23. Its residuals are well-ordered and it is reset-stable, as after any reset we necessarily produce an word in W. Therefore, W is positional.

We remark that this objective is not bipositional, as $\mathsf{Res}(\Sigma^\omega \setminus W)$ is not well-founded. This contrast with the case of ω-regular open objectives, for which all positional open objectives are bipositional (Corollary 4.10). ◆

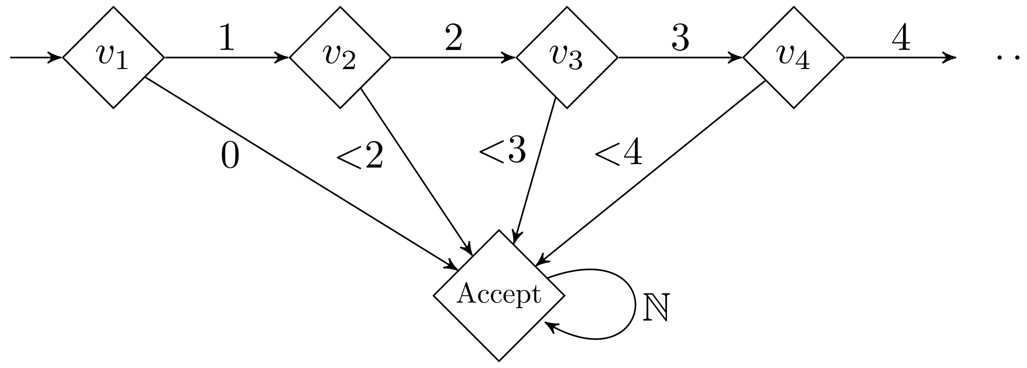

Figure 23. Automaton recognising the objective W from Example 8.12.

8.3 1-to-2-player lift and addition of neutral letters

COROLLARY 8.13 (1-to-2-player lift for open and closed objectives). *Let $W \subseteq \Sigma^\omega$ be an open or closed objective. If W is positional over ε-free Eve-games, then W is positional over all game graphs.*

We also obtain from our proofs that the Neutral letter conjecture (Conjecture 3.8) holds for open and closed objectives.

COROLLARY 8.14 (Closure under addition of neutral letters). *Let $W \subseteq \Sigma^\omega$ be an open or closed objective. If W is positional, then W^ε is positional.*

PROOF. In the proof of Theorems 8.4 and 8.10, we obtained the positionality of objectives by providing well-ordered monotone (κ, W)-universal graphs. Proposition 3.7 allows us to conclude. ■

We have therefore obtained the 1-to-2-player lift for ω-regular objectives, as well as open and closed ones. However, the 1-to-2-player lift does not hold for arbitrary objectives. A counter-example appears in [31, Section 7], moreover, the objective presented there is in Σ_2^0, that is, a countable union of closed objectives. Another counter-example, discussed in [34] and [55, p.236][12] is:

$$\mathbf{MP}^{\mathbb{Q}} = \{w \in \{0,1\}^\omega \mid \liminf_n \frac{1}{n} \sum_{i=0}^{n} w_i \quad \text{is rational}\}.$$

We provide here yet a different example, which is conceptually simpler but of higher topological complexity than the one in [31].

PROPOSITION 8.15 (No general 1-to-2-player lift). *There is an objective $W \subseteq \Sigma^\omega$ that is positional over Eve-games, but is not positional over all game graphs.*

12 In his PhD [55], Vandenhove discusses the 1-to-2-player lift for finite game graphs, but this counter-example also applies to infinite game graphs.

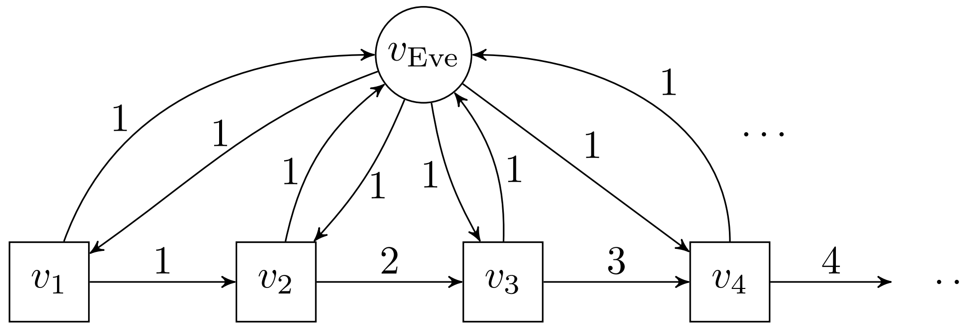

Figure 24. Game in which Eve wins if only finitely many letters are produced infinitely often. She can win by sending the token further and further away, but she cannot win positionally.

PROOF. Let Σ be any infinite alphabet, and let W be the following objective:

$$W_{\text{fin}} = \{w \in \Sigma^\omega \mid \text{ the set of letters occurring infinitely often in } w \text{ is finite}\}.$$

To show that it is not positional, we consider the game in Figure 24, in which Eve controls a single vertex, from which she can send the token to any vertex v_i controlled by Adam. We claim that Eve can win this game from every vertex by using the following strategy: she keeps track of the maximal index $i_{\max}$ such that the play has passed trough vertex $v_{i_{\max}}$. Whenever Adam sends back the token to v_{Eve}, she will go to vertex $v_{i_{\max}}$. This strategy ensures that only letter 1 will be produced infinitely often.

However, Eve cannot win using a positional (or even finite memory) strategy. Such a strategy will only consider finitely many edges $v_{\text{Eve}} \xrightarrow{1} v_i$. Let v_k be the maximal such vertex. Adam can win against such strategy by producing longer and longer paths, and then sending back the token to the vertex controlled by Eve: $v_i \overset{i(i+1)\dots}{\rightsquigarrow} v_{k+j} \xrightarrow{1} v_{\text{Eve}}$. In this way, all numbers greater that k will be produced infinitely often.

We show that W_{fin} is positional over Eve-games.

CLAIM 8.16. *For every Eve-game $\mathcal{G}$ with winning condition W_{fin} and every fixed vertex v_0, if Eve wins from v_0, she can win from v_0 using a positional strategy.*

Proof. Assume that Eve wins from v_0. A strategy from v_0 is just an infinite path from v. Consider such a path. If this path does not visit a same vertex twice, it is already a positional strategy. On the contrary, let v_k be the first vertex that repeats. Consider the first two occurrences of v_{rep}:

$$v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots v_k \xrightarrow{a_k} \cdots v_{k+j} \xrightarrow{a_{k+j}} v_k.$$

Then, the positional strategy indicating to take the edge $v_i \xrightarrow{a_i} v_{i+1}$ for $i \leq k+j$ is winning. ◆

We use this claim to prove that W_{fin} is (uniformly) positional over Eve-games. Let $\mathcal{G}$ be an Eve-game with winning condition W_{fin}. By prefix-independence of W_{fin}, we can suppose without loss of generality that Eve wins from every vertex in $\mathcal{G}$. Let $v_1, v_2, \dots$ a (potentially transfinite) enumeration of vertices in $\mathcal{G}$. We will define a positional strategy strat: $V \to E$ by transfinite induction. At step λ, let $\mathcal{G}_\lambda$ be the game obtained by removing vertices for which strat has already been defined. Let i be the minimal index such that v_i appears in $\mathcal{G}_\lambda$. By the previous claim, Eve has a positional strategy in $\mathcal{G}_\lambda$ that wins from v_i. Let V_λ be the vertices reachable from v by using this strategy, and for $v \in V_\lambda$ let strat(v) be the edge indicated by such strategy. We let V'_λ be the vertices in $\mathcal{G}_\lambda \setminus V_\lambda$ from which Eve can reach V_λ, and fix a positional strategy doing so. We let strat(v') being given by this strategy for these vertices. It is immediate that strat is a positional strategy in $\mathcal{G}$ that wins from all vertices. ■

9. Conclusions

The results presented in Section 3 effectively address most open questions regarding positionality in the context of ω-regular languages. Yet, it would be reasonable to seek a "better" characterisation. One drawback of our approach is its conceptual complexity and its exclusive focus on automata; the characterisation is based on syntactic and combinatorial properties of parity automata, rather than on intrinsic language-theoretical properties of the languages they recognise. In this respect, the insights gained about positionality are somewhat limited. Therefore, we believe that there is still room for improvement and for a deeper understanding of the class of positional ω-regular objectives.

To conclude, we discuss further research directions extending our results. We start by discussing the follow-up work [22], generalising the results of this paper to objectives requiring memory.

9.1 Follow-up work: Positionality and memory of $\mathcal{BC}(\Sigma_2^0)$ languages

There are two natural orthogonal directions to extend our results: to consider broader classes of objectives and to characterise their memory requirements rather than just positionality.

In Section 8 we have presented positionality results for some non-ω-regular objectives, namely for closed and open objectives. Going further in that direction, the goal is to develop characterisations for more complex objectives defined by topological properties, mainly, higher classes in the Borel hierarchy. A natural first step is to look at objectives in $\mathcal{BC}(\Sigma_2^0)$: the class of boolean combinations of objectives in Σ_2^0 (countable unions of closed objectives), or equivalently, objectives recognised by infinite deterministic parity automata [53, Sect. 5]. In [22], we provide a characterisation of objectives in $\mathcal{BC}(\Sigma_2^0)$ for which Eve can play optimally with $\leq k$ states of memory. In particular, for $k = 1$, this characterises positionality, allowing us to prove Kopczyński's conjecture (Conjecture 1.1) for the class of $\mathcal{BC}(\Sigma_2^0)$ objectives. The simplest class in the Borel hierarchy for which we do not have a satisfactory characterisation of positionality is $\Delta_3^0 = \Sigma_3^0 \cap \Pi_3^0$, for which we do not know whether Kopczyński's conjecture holds.

Concerning 1-to-2-player lifts for positionality (for one player), the results from this paper are closed to the theoretical limit. Indeed, we have shown that the 1-to-2-player lift holds for ω-regular, open and closed objectives (Theorem 3.3 and Corollary 8.13), and it is known that no such lift holds for Σ_2^0-objectives [31, Section 7].

9.2 Minimisation and canonisation of parity automata

For the proof of Theorem 3.1 (Section 5.2), we have introduced new notions concerning congruences for parity automata, as well as different transformations of automata. This transformations exhibit a flavour similar to what one might expect from minimisation algorithms, as their main purpose is to remove redundant states from automata. We believe that this techniques will be valuable for the study of ω-automata in other contexts, as they allow for a fine analysis of the structure of the automata.

Also, a key ingredient in our proof is the use of history-deterministic automata, in particular, a generalisation of the minimisation algorithm for history-deterministic coBüchi automata introduced by Abu Radi and Kupferman [1]. We see this fact as further evidence that history-deterministic automata provide canonicity properties.

In fact, we can derive actual concrete statements about the minimisation of automata from our results:

PROPOSITION 9.1. *Given a deterministic parity automata recognising a bipositional language L, we can compute in polynomial time the size of a minimal deterministic (resp. history-deterministic) parity automata for L.*

PROOF. A necessary condition for a language L to be bipositional is that it must be recognised by a parity automaton on top of the automaton of residuals (Theorem 7.1). Therefore, a minimal deterministic or HD automaton for L will have as many states as residuals of the language. To compute the number of residuals of the language, it suffices to determine the number of equivalence classes of states recognising the same language in the input deterministic automaton. For this, it suffices to do an equivalence check for each pair of states, and each of them takes polynomial time [23]. ∎

PROPOSITION 9.2. *Deterministic (resp. history-deterministic) Büchi and coBüchi automata recognising positional languages can be minimised in polynomial time.*

PROOF. In the case of Büchi automata, a positional language can be recognised by the automaton of residuals (Propositions 4.17), which is necessarily minimal amongst history-deterministic automata.

In the case of coBüchi automata, we obtained that the minimal history-deterministic automaton of Abu Radi and Kupferman, computable in polynomial time, can be taken deterministic for positional languages (Section 4.4). ∎

We conjecture that our methods may lead to similar results in the more general case of parity automata, and that minimal automata for positional languages can be obtained just by merging states of signature automata.

CONJECTURE 9.3. *Deterministic and history-deterministic parity automata recognising positional languages can be minimised in polynomial time. Moreover, history-deterministic parity automata for this class of languages are not more succinct than deterministic ones.*

9.3 Algorithms for ω-regular games

A major algorithmic problem is the resolution of games on graphs, that is, deciding whether Eve has a winning strategy in a given game. A common approach to solving games with ω-regular objectives is to reduce them to parity games: Build the product between the game graph and a parity automaton recognising the objective, and then apply a parity game solver to it. Using state-of-the-art quasipolynomial-time parity game solvers [15], this leads to a complexity of roughly $(|\mathcal{G}||\mathcal{A}|)^{\log d}$, where d is the number of priorities used by $\mathcal{A}$. As discussed in the previous section, if $\mathcal{A}$ recognises a positional language, we have provided a polynomial-time algorithm reducing its size (to possibly a minimal one).

Moreover, if the objective $W = \mathcal{L}(\mathcal{A})$ is positional, an alternative method is to apply a value iteration algorithm directly on the game. For this, we need a $(|G|, W)$-universal graph $\mathcal{U}$ (which exists by [47, Theorem 3.1]), which leads to an algorithm working in time $O(|G||\mathcal{U}|)$ [24]. Moreover, one of the appealing aspects of this approach is the possibility to establish lower bounds for the size of universal graphs, which has been succesfully used in the past for setting the limits of various families of algorithms [28, 24].

In this work, we have shown how to obtain a universal graph from a signature automaton. A possible research direction would be to optimize this construction and determine the size of universal graphs for positional ω-regular objectives. In this way, we could expect to:

— decrease the complexity of solving positional ω-regular games to about $|G|^{(1+\log d)}|\mathcal{A}|$,

— obtain matching lower bounds for $(n, \mathcal{L}(\mathcal{A}))$-universal graphs.

References

[1] **Bader Abu Radi and Orna Kupferman**. Minimization and canonization of GFG transition-based automata. *Log. Methods Comput. Sci.* 18(3), 2022. DOI (5, 33, 36, 46, 50, 82)

[2] **André Arnold**. A syntactic congruence for rational omega-language. *Theor. Comput. Sci.* 39:333–335, 1985. DOI (5)

[3] **Alessandro Bianco, Marco Faella, Fabio Mogavero, and Aniello Murano**. Exploring the boundary of half-positionality. *Ann. Math. Artif. Intell.* 62(1-2):55–77, 2011. DOI (2, 27, 41)

[4] **Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann**. Graph games and reactive synthesis. **Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem**, editors, *Handbook of Model Checking*, pages 921–962. Springer International Publishing, 2018. DOI (2)

[5] **León Bohn and Christof Löding**. Constructing deterministic parity automata from positive and negative examples. *TheoretiCS*, 3, 2024. DOI (5)

[6] **Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak**. Nondeterminism in the presence of a diverse or unknown future. *International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, pages 89–100, 2013. DOI (97)

[7] **Udi Boker and Karoliina Lehtinen**. When a little nondeterminism goes a long way: an introduction to history-determinism. *ACM SIGLOG News*, 10(1):24–51, 2023. DOI (5, 14)

[8] **Patricia Bouyer, Antonio Casares, Mickael Randour, and Pierre Vandenhove**. Half-positional objectives recognized by deterministic büchi automata. *Log. Methods Comput. Sci.* 20(3), 2024. DOI (3, 4, 27, 29, 64)

[9] **Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba**. Infinite runs in weighted timed automata with energy constraints. *International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2008)*, volume 5215 of *Lecture Notes in Computer Science*, pages 33–47, 2008. DOI (70)

[10] **Patricia Bouyer, Nathanaël Fijalkow, Mickael Randour, and Pierre Vandenhove**. How to play optimally for regular objectives? *International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *LIPIcs*, 118:1–118:18, 2023. DOI (2)

[11] **Patricia Bouyer, Mickael Randour, and Pierre Vandenhove**. Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs. *TheoretiCS*, 2, 2023. DOI (2, 3, 70–72)

[12] **Patricia Bouyer, Stéphane Le Roux, Youssouf Oualhadj, Mickael Randour, and Pierre Vandenhove**. Games where you can play optimally with arena-independent finite memory. *International Conference on Concurrency Theory (CONCUR 2020)*, volume 171, 24:1–24:22, 2020. DOI (3)

[13] **J. Richard Büchi**. On a decision method in restricted second order arithmetic. *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science*:1–11, 1962. (3)

[14] **J. Richard Büchi and Lawrence H. Landweber**. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969. URL (2)

[15] **Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan**. Deciding parity games in quasi-polynomial time. *SIAM Journal on Computing*, 51(2):STOC17-152-STOC17–188, 2022. DOI (83)

[16] **Olivier Carton and Ramón Maceiras**. Computing the Rabin index of a parity automaton. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*:495–506, 1999. DOI (14)

[17] **Antonio Casares**. On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions. *Computer Science Logic (CSL 2022)*, volume 216, 12:1–12:17, 2022. DOI (3)

[18] **Antonio Casares**. Structural properties of automata over infinite words and memory for games (Propriétés structurelles des automates sur les mots infinis et mémoire pour les jeux). PhD Thesis, Université de Bordeaux, France, 2023. URL (12)

[19] **Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen**. From Muller to Parity and Rabin Automata: Optimal Transformations Preserving (History) Determinism. *TheoretiCS*, Volume 3, 2024. DOI (5, 14)

[20] **Antonio Casares and Pierre Ohlmann**. Characterising memory in infinite games. *Logical Methods in Computer Science*, Volume 21, Issue 1, 28, 2025. DOI (4, 10, 73)

[21] **Antonio Casares and Pierre Ohlmann**. Positional ω-regular languages. *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2024)*, 21:1–21:14. ACM, 2024. DOI (1)

[22] **Antonio Casares and Pierre Ohlmann**. The memory of ω-regular and $BC(\Sigma_0^2)$ objectives. *International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *LIPIcs*, 149:1–149:18, 2025. DOI (81)

[23] **Edmund M. Clarke, I. A. Draghicescu, and Robert P. Kurshan**. A unified approch for showing language inclusion and equivalence between various types of omega-automata. *Inf. Process. Lett.* 46(6):301–308, 1993. DOI (64, 66, 82)

[24] **Thomas Colcombet, Nathanaël Fijalkow, Pawel Gawrychowski, and Pierre Ohlmann**. The theory of universal graphs for infinite duration games. *Log. Methods Comput. Sci.* 18(3), 2022. DOI (66, 83)

[25] **Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn**. Playing safe. *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, volume 29, pages 379–390, 2014. DOI (3, 24, 73)

[26] **Thomas Colcombet, Nathanaël Fijalkow, and Florian Horn**. Playing safe, ten years later. *Log. Methods Comput. Sci.* 20(1), 2024. DOI (2)

[27] **Thomas Colcombet and Damian Niwiński**. On the positional determinacy of edge-labeled games. *Theor. Comput. Sci.* 352(1-3):190–196, 2006. DOI (2, 5, 70)

[28] **Wojciech Czerwinski, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdzinski, Ranko Lazic, and Pawel Parys**. Universal trees grow inside separating automata: quasi-polynomial lower bounds for parity games. *ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, pages 2333–2349. SIAM, 2019. DOI (83)

[29] **Rüdiger Ehlers and Sven Schewe**. Natural colors of infinite words. *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*, volume 250, 36:1–36:17, 2022. DOI (5)

[30] **E. Allen Emerson and Charanjit S. Jutla**. Tree automata, mu-calculus and determinacy (extended abstract). *IEEE Symposium on Foundations of Computer Science (FOCS 1991)*, pages 368–377, 1991. DOI (2, 10, 27)

[31] **Hugo Gimbert and Edon Kelmendi**. Two-player perfect-information shift-invariant submixing stochastic games are half-positional. *CoRR*, abs/1401.6575, 2014. version 2. URL (79, 82)

[32] **Hugo Gimbert and Wieslaw Zielonka**. Games where you can play optimally without any memory. *International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653, pages 428–442, 2005. DOI (2, 3, 71)

[33] **Nils Klarlund**. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2):243–268, 1994. DOI (2)

[34] **Eryk Kopczyński**. Half-positional Determinacy of Infinite Games. PhD thesis, University of Warsaw, 2008. (2, 4, 22, 27, 35, 36, 79)

[35] **Eryk Kopczyński**. Half-positional determinacy of infinite games. *International Colloquium on Automata, Languages, and Programming (ICALP 2006)*, pages 336–347, 2006. DOI (4)

[36] **Eryk Kopczyński**. Omega-regular half-positional winning conditions. *Computer Science Logic (CSL 2007)*, volume 4646 of *Lecture Notes in Computer Science*, pages 41–53, 2007. DOI (3)

[37] **Alexander Kozachinskiy**. Energy games over totally ordered groups. *Computer Science Logic (CSL 2024)*, volume 288, 34:1–34:12, 2024. DOI (3, 4, 70)

[38] **Alexander Kozachinskiy**. Infinite separation between general and chromatic memory. *Latin American Symposium on Theoretical Informatics (LATIN 2024)*, volume 14579 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2024. DOI (3)

[39] **Alexander Kozachinskiy**. One-to-two-player lifting for mildly growing memory. *Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *LIPIcs*, 43:1–43:21, 2022. DOI (3)

[40] **Alexander Kozachinskiy**. State complexity of chromatic memory in infinite-duration games. *CoRR*, abs/2201.09297, 2022. (3)

[41] **Denis Kuperberg and Michał Skrzypczak**. On determinisation of good-for-games automata. *International Colloquium on Automata, Languages, and Programming (ICALP 2015)*, pages 299–310, 2015. DOI (5, 18, 34)

[42] **Orna Kupferman**. Using the past for resolving the future. *Frontiers Comput. Sci.* 4, 2022. DOI (5)

[43] **Oded Maler and Ludwig Staiger**. On syntactic congruences for ω-languages. *Theoretical Computer Science*, 183(1):93–112, 1997. Formal Language Theory. DOI (5)

[44] **Donald A. Martin**. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975. URL (8)

[45] **Robert McNaughton**. Testing and generating infinite sequences by a finite automaton. *Information and control*, 9(5):521–530, 1966. DOI (3)

[46] **Andrzej W. Mostowski**. Regular expressions for infinite trees and a standard form of automata. *Symposium on Computability Theory (SCT 1984)*, pages 157–168, 1984. DOI (3, 13)

[47] **Pierre Ohlmann**. Characterizing positionality in games of infinite duration over infinite graphs. *TheoretiCS*, 2, 2023. DOI (3, 9, 10, 22, 56, 83)

[48] **Pierre Ohlmann**. Monotonic graphs for parity and mean-payoff games. (Graphes monotones pour jeux de parité et à paiement moyen). PhD thesis, Université Paris Cité, France, 2021. URL (3)

[49] **Pierre Ohlmann and Michal Skrzypczak**. Positionality in Σ^0_2 and a completeness result. *Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289, 54:1–54:18, 2024. DOI (3, 4)

[50] **Bader Abu Radi and Orna Kupferman**. On semantically-deterministic automata. *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, 109:1–109:20, 2023. DOI (18)

[51] **Bertrand Le Saëc**. Saturating right congruences. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 24:545–559, 1990. DOI (5)

[52] **Robert S. Streett and E. Allen Emerson**. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81:249–264, 1989. DOI (10)

[53] **Wolfgang Thomas**. Automata on infinite objects. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics*, 1991. URL (24, 81)

[54] **Wolfgang Thomas**. On the synthesis of strategies in infinite games. *Symposium on Theoretical Aspects of Computer Science (STACS 1995)*, pages 1–13, 1995. DOI (2)

[55] **Pierre Vandenhove**. Strategy complexity of zero-sum games on graphs. (Complexité des stratégies des jeux sur graphes à somme nulle). PhD thesis, University of Mons, Belgium, 2023. URL (3, 4, 79)

[56] **Igor Walukiewicz**. Pushdown processes: games and model checking. *International Conference on Computer Aided Verification (CAV 1996)*, volume 1102 of *Lecture Notes in Computer Science*, pages 62–74, 1996. DOI (10)

[57] **Wieslaw Zielonka**. An invitation to play. *Mathematical Foundations of Computer Science (MFCS 2005)*, volume 3618, pages 58–70, 2005. DOI (71)

[58] **Wiesław Zielonka**. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998. DOI (4)

A. Full proofs for Section 5.2

In this Appendix, we include full proofs for all the propositions and lemmas appearing in Section 5.2. To help formalise them, we first introduce some technical artillery that will come in handy to deal with the structure of total preorders of signature automata. We begin by introducing nice transformations of automata equipped with a priority-faithful relation in Section A.1. Then, in Section A.2 we give the full details of the induction constructing a structured signature automaton for a positional language.

A.1 Nice transformations of automata

To recursively build a structured signature automaton, we will apply a sequence of transformations to a given d-signature automaton $\mathcal{A}$, by removing states and adding or redirecting edges in such a way that relations $\sim_x$ are preserved in a strong sense formalised in this section.

GLOBAL HYPOTHESIS. We recall that automata are assumed to be complete and semantically deterministic.

Automata with a common subautomaton Let $\mathcal{A}$ be a semantically deterministic automaton with states $Q_\mathcal{A}$, and let $\sim_\mathcal{A}$ be the congruence given by the equality of residuals. We note that if $\mathcal{B}$ is an automaton with states $Q_\mathcal{B} \subseteq Q_\mathcal{A}$, relation $\sim_\mathcal{A}$ induces an equivalence relation over $Q_\mathcal{B}$ (which, in general, is not a congruence nor coincides with the equality of residuals of $\mathcal{B}$).

LEMMA A.1 (Automata preserving the structure of residuals). *Let $\mathcal{A}$ be a semantically deterministic parity automaton with states $Q_\mathcal{A}$ and let $\mathcal{B}$ be a parity automaton with states $Q_\mathcal{B} \subseteq Q_\mathcal{A}$. Assume that $\sim_\mathcal{A}$ is a congruence over $\mathcal{B}$ and that $\mathcal{B}/_{\sim_\mathcal{A}} = \mathcal{A}/_{\sim_\mathcal{A}}$. Let $\mathcal{A}'$ be a subautomaton of both $\mathcal{A}$ and $\mathcal{B}$. If a word $w \in \Sigma^\omega$ admits an accepting run in $\mathcal{B}$ that eventually remains in $\mathcal{A}'$, then w is also accepted by $\mathcal{A}$.*

PROOF. Let

$$q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} q_2 \xrightarrow{w_2} \dots \xrightarrow{w_{k-1}} q_k \xrightarrow{w_k} q_{k+1} \xrightarrow{w_{k+1}} \dots$$

be an accepting run over w in $\mathcal{B}$ such that the suffix from q_k is contained in $\mathcal{A}'$ (meaning that both the states and the transitions used are part of $\mathcal{A}'$). We consider the projection of the prefix of size k of the run in the quotient automaton $\mathcal{B}/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. By Lemma 2.14, there is a run over $w_0 \dots w_{k-1}$ in $\mathcal{A}$, $p_0 \xrightarrow{w_0} p_1 \xrightarrow{w_1} \dots p_k$ whose projection over the quotient automaton coincides with the previous one. Therefore, $p_k \sim_{\mathcal{A}} q_k$, that is, $\mathcal{L}(\mathcal{A}_{p_k}) = \mathcal{L}(\mathcal{A}_{q_k})$. Since $w_{k+1}w_{k+2}\dots$ admits an accepting run from q_k in $\mathcal{A}$, it also admits an accepting run from p_k, and w is accepted by $\mathcal{A}$. ■

Nice transformations of automata For $x \in \mathbb{N}$, we denote by $\mathcal{A}|_{\geq x}$ the subautomaton of $\mathcal{A}$ induced by the set of transitions using a priority $\geq x$.

DEFINITION A.2 (Nice transformation at level x). Let $\mathcal{A}$ be a semantically deterministic parity automaton over Σ with states Q, let x be a priority, and let $\sim$ be a $[0, x-1]$-faithful congruence over $\mathcal{A}$. Let $\mathcal{A}'$ be a parity automaton over Σ with states $Q' \subseteq Q$. We denote $\sim$ the induced relation over Q'. We say that $\mathcal{A}'$ is a *$\sim$-nice transformation of $\mathcal{A}$ at level x* if:

- $\sim$ is a $[0, x-1]$-faithful congruence over $\mathcal{A}'$ and $\mathcal{A}/^{\sim}_{\leq x-1} = \mathcal{A}'/^{\sim}_{\leq x-1}$ (see Definition 5.4),
- $\sim_{\mathcal{A}}$ is a congruence over $\mathcal{A}'$ and $\mathcal{A}'/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$, and
- $\mathcal{A}'|_{\geq x+1}$ coincides with the subautomaton of $\mathcal{A}|_{\geq x+1}$ induced by the states in Q'.

Intuitively, if $\mathcal{A}'$ is a nice transformation of $\mathcal{A}$ at level x, it means that the only relevant modifications applied to $\mathcal{A}$ concern x-transitions. The structure of the quotient automaton for priorities $<x$ is left unchanged, and so is the acceptance of runs that eventually only produce priorities $>x$.

REMARK A.3. We note that if $\sim$ is an equivalence relation that refines $\sim_{\mathcal{A}}$, then the second item of Definition A.2 is implied by the first one. This is in particular the case if $\sim$ is an equivalence relation $\sim_x$ of a d-signature automaton, for $0 \leq x \leq d$.

LEMMA A.4 (Preservation of classes and priorities in nice transformations). *Let $\mathcal{A}$ be a semantically deterministic parity automaton equipped with a $[0, x-1]$-faithful congruence $\sim$. Suppose that $\mathcal{A}$ is deterministic over $>x$-transitions. Let $\mathcal{A}'$ be a $\sim$-nice transformation of $\mathcal{A}$ at level x. If $q \sim q'$ are two states of $\mathcal{A}'$ such that there is path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$, then a path $q' \overset{w:y'}{\rightsquigarrow} p'$ in $\mathcal{A}'$ satisfies:*

- *$p \sim p'$,*
- *if $y < x$, then $y' = y$, and*
- *if $y \geq x$, then $y' \geq x$.*

PROOF. The equivalence $p \sim p'$ follows from the fact that $\sim$ is a congruence in $\mathcal{A}'$ and $\mathcal{A}'/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. For $y \leq x-1$ or $y' \leq x-1$, the equality $y' = y$ follows from the equality of the $\leq(x-1)$-quotient automaton. This directly implies the third item. ■

LEMMA A.5. *Let $\mathcal{A}$ be a ε-completion that admits a $[0,x]$-faithful congruence $\sim$. If a word $w \in \Sigma^\omega$ admits a run such that the minimal priority produced infinitely often is $y \leq x$, then the minimal priority produced infinitely often by any run over w is y. In particular, if y is odd, w is rejected with priority y.*

PROOF. Let $q_0 \xrightarrow{w_1:y_1} q_1 \xrightarrow{w_2:y_2} \dots$ and $q'_0 = q_0 \xrightarrow{w_1:y'_1} q'_1 \xrightarrow{w_2:y'_2} \dots$ be two runs over the same word in $\mathcal{A}$ ($q_0 = q'_0$ being the initial state of $\mathcal{A}$). Since $\sim$ is a congruence, we obtain by induction that $q_i \sim q'_i$ for every i. Moreover, as, for $y \leq x$, y-transitions act uniformly over $\sim$-classes, each time that $y_i \leq x$, we have that $y'_i = x_i$. ∎

LEMMA A.6 (Nice transformations preserve acceptance of most words). *Let $\mathcal{A}$ be a semantically deterministic parity automaton equipped with a $[0,x-1]$-faithful congruence $\sim$. Let $\mathcal{A}'$ be a $\sim$-nice transformation of $\mathcal{A}$ at level x. We have:*

- *A word $w \in \Sigma^\omega$ can be accepted with an even priority $y < x$ in $\mathcal{A}'$ if and only if w can be accepted with priority y in $\mathcal{A}$.*
- *A word $w \in \Sigma^\omega$ is rejected with an odd priority $y < x$ in $\mathcal{A}'$ if and only if w is rejected with priority y in $\mathcal{A}$.*
- *If a word $w \in \Sigma^\omega$ can be accepted with an even priority $y > x$ in $\mathcal{A}'$, then it is accepted by $\mathcal{A}$.*

If moreover $\mathcal{A}$ is homogeneous and deterministic over transitions using priorities $> x$, we have:

- *If there is a rejecting run over $w \in \Sigma^\omega$ in $\mathcal{A}'$ producing as minimal priority $y > x$, then w is rejected by $\mathcal{A}$.*

PROOF. Let $w \in \Sigma^\omega$ be a word accepted with a priority $y < x$ in $\mathcal{A}$ (resp. $\mathcal{A}'$). Then, by Lemma 5.5, w is accepted by the quotient automaton $\mathcal{A}/_{\sim_{\leq x-1}} = \mathcal{A}'/_{\sim_{\leq x-1}}$. Again by Lemma 5.5, w is accepted by $\mathcal{A}'$ (resp. $\mathcal{A}$). The second item is obtained using the same argument, combined with Lemma A.5.

The third item is directly implied by Lemma A.1, as $\mathcal{A}'|_{\geq x+1}$ is a subautomaton of $\mathcal{A}|_{\geq x+1}$.

For the last item, let $q_0 \overset{w_0}{\rightsquigarrow} q' \overset{w'}{\rightsquigarrow}$ be a rejecting run over w in $\mathcal{A}'$ such that the suffix $q' \overset{w'}{\rightsquigarrow}$ does not produce any priority $\leq x$ (that is, it is contained in $\mathcal{A}'|_{\geq x+1}$). By determinism and homogeneity, this is the only run over w' from q' in $\mathcal{A}$, and therefore $w' \notin \mathcal{L}(\mathcal{A}_{q'})$. We conclude using the equality $\mathcal{A}'/_{\sim_{\mathcal{A}}} = \mathcal{A}/_{\sim_{\mathcal{A}}}$. ∎

A.2 From positionality to structured signature automata: Full proofs for Section 5.2

We provide all the technical details for the proofs of the propositions appearing in Section 5.2. We first state some useful simple lemmas.

LEMMA A.7. *Let $W \subseteq \Sigma^\omega$ be positional over finite, ε-free Eve-games. Then, for every word $u \in \Sigma^*$, objective $u^{-1}W$ is positional over finite, ε-free Eve-games.*

PROOF. Any game with vertices V witnessing non-positionality of $u^{-1}W$ can be turned into a game witnessing non-positionality of W by adding, for every $v \in V$, a fresh vertex v_u and a path $v_u \overset{u}{\rightsquigarrow} v$. ■

LEMMA A.8. *To check if a parity automaton is in normal form, it suffices to verify that, if q and p are in a same positive (resp. negative) SCC and there is a transition $q \xrightarrow{x} p$ producing priority $x > 0$ (resp. $x > 1$), then there are two paths $p \rightsquigarrow q$ producing as minimal priority x and $x - 1$, respectively.*

PROOF. We do the proof for the case of a positive SCC. Assume that there is a path $q = q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} \dots \xrightarrow{x_n} q_n = p$ with $x = \min x_i > 0$. By hypothesis, for $1 \leq i \leq n$, there are paths $q_i \overset{x_i - 1}{\rightsquigarrow} q_{i-1}$. Concatenating them, we obtain a path $p \overset{x-1}{\rightsquigarrow} q$. Iterating this process, we can obtain loops $q \rightsquigarrow p \rightsquigarrow q$ producing as minimal priority any number in $[0, x-1]$. To obtain a path $p \overset{x}{\rightsquigarrow} q$, we just use the existence of paths $q_i \overset{x_i}{\rightsquigarrow} q_{i-1}$. ■

A.2.1 Base case: Total order of residual classes

LEMMA A.9 (Total order of residual classes). *Let $W \subseteq \Sigma^\omega$ be an ω-regular objective that is positional over finite, ε-free Eve-games. Then,* $\mathsf{Res}(W)$ *is totally ordered by inclusion.*

PROOF. The proof is almost identical to that of Lemma 4.1. We show the contrapositive. Assume that W has two incomparable residuals, $u_1{}^{-1}W$ and $u_2{}^{-1}W$. We consider first the case $u_1 \neq \varepsilon$ and $u_2 \neq \varepsilon$. Take $w_1 \in u_1^{-1}W \setminus u_2^{-1}W$ and $w_2 \in u_2^{-1}W \setminus u_1^{-1}W$. Thanks to ω-regularity, we can take these words of the form $w_i = u_i'(w_i')^\omega$, with $u_i', w_i' \in \Sigma^+$, for $i = 1, 2$. We have

$$u_1 w_1 \in W, \quad u_1 w_2 \notin W,$$
$$u_2 w_1 \notin W, \quad u_2 w_2 \in W.$$

Consider the ε-free, finite, Eve-game $\mathcal{G}$ represented in Figure 25. Eve wins $\mathcal{G}$ from v_1 and v_2: if a play starts in v_i, for $i = 1, 2$, she just has to take the path labelled $u_i'(w_i')^\omega$ from v_{choice}. However, she cannot win from both v_1 and v_2 using a positional strategy. Indeed, such positional strategy would choose one path $v_{\mathsf{choice}} \overset{u_i(w_i')}{\rightsquigarrow}$, and the play induced when starting from v_{1-i} would be losing.

Finally, we take care of the case in which $[u_1] = \{\varepsilon\}$ (symmetric for u_2). In that case, we cannot take $u_1 \neq \varepsilon$. We remark that, since $[u_1] \neq [u_2]$, we can take $u_2 \neq \varepsilon$. We consider the game from Figure 25 in which we simply remove vertex v_1. This game is ε-free, and Eve can win from both v_2 and v_{choice}, but not positionally. ■

GLOBAL HYPOTHESIS. In all the rest of the subsection, we assume that $x \geq 2$.

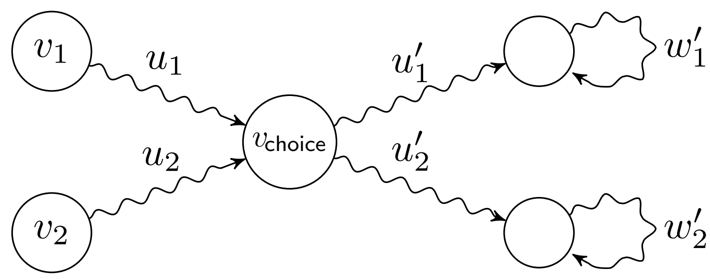

Figure 25. A game $\mathcal{G}$ in which Eve cannot play optimally using positional strategies if $\mathsf{Res}(W)$ is not totally ordered, as in the proof of Lemma A.9.

A.2.2 Safe centrality and relation $\sim_{x-1}$. Proof of Lemma 5.16

In this paragraph we give a proof of Lemma 5.16. We assume that $x \geq 2$ is an even priority and $\mathcal{A}$ is a deterministic $(x-2)$-structured signature automaton with initial state q_{init}.

LEMMA 5.16 ((<x)-safe centralisation). (Restated) *There exists a $(x-2)$-structured signature automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ which is:*

- *deterministic over transitions with priority different from $x-1$,*
- *homogeneous,*
- *history-deterministic, and*
- *$(<x)$-safe centralised.*

Moreover, $\mathcal{A}'$ can be obtained in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

Hypothesis. During the proof, we will lose the determinism of $\mathcal{A}$. However, in all the subsection we will maintain the three first required properties. In the statements of all lemmas, $\mathcal{A}$ will stand for an $(x-2)$-structured signature automaton that is:

- deterministic over transitions with priority different from $x-1$,
- homogeneous, and
- history-deterministic.

Saturation. We say that an automaton $\mathcal{A}'$ is $(x-1)$-*saturated* if for every state q and letter $a \in \Sigma$, if a transition $q \xrightarrow{a:x-1} p$ exists in $\mathcal{A}'$, then $q \xrightarrow{a:x-1} p'$ appears in $\mathcal{A}'$ for all $p' \sim_{x-2} p$.[13] The $(x-1)$-saturation of $\mathcal{A}$ is the automaton obtained by adding all those transitions.

REMARK A.10. The $(x-1)$-saturation of $\mathcal{A}$ is homogeneous and deterministic over transitions with priority different from $x-1$.

13 We note that this definition slightly differs from the definition of 1-saturation used in the warm-up (Section 4.4). In particular, the definition of the warm-up does not preserve homogeneity. We allow ourselves these small disagreements of definitions for the sake of clarity in the presentation in each respective subsection.

LEMMA A.11. *The $(x-1)$-saturation of $\mathcal{A}$ is in normal form.*

PROOF. We use the characterisation of normal form given in Lemma A.8. Let $\mathcal{A}'$ be the $(x-1)$-saturation of $\mathcal{A}$. The property of Lemma A.8 is satisfied for transitions already appearing in $\mathcal{A}$, as $\mathcal{A}$ is assumed to be in normal form. Let $q \xrightarrow{a:x-1} p$ be a transition added by the saturation process, and let $q \xrightarrow{a:x-1} p'$ be a transition in $\mathcal{A}$ with $p \sim_{x-2} p'$. By Item 5 of the definition of structured signature automaton, there is a path $p \overset{u:>x-2}{\rightsquigarrow} p'$ in $\mathcal{A}$. By normality of $\mathcal{A}$, there are also paths $p' \overset{u_1:x-1}{\rightsquigarrow} q$ and $p' \overset{u_2:x-2}{\rightsquigarrow} q$. We obtain the two desired paths in $\mathcal{A}'$:

$$p \overset{u:>x-2}{\rightsquigarrow} p' \overset{u_1:x-1}{\rightsquigarrow} q\,, \quad \text{and} \quad p \overset{u:>x-2}{\rightsquigarrow} p' \overset{u_2:x-2}{\rightsquigarrow} q.$$ ■

The following lemma states that $(x-1)$-saturation is a $\sim_{x-2}$-nice transformation at level $x-1$, so Lemmas A.6 and A.4 can be applied. We recall that $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$, so congruence of $\sim_{\mathcal{A}}$ is implied by that of $\sim_{x-2}$.

LEMMA A.12. *The $(x-1)$-saturation of $\mathcal{A}$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.*

PROOF. Let $\mathcal{A}'$ be the $(x-1)$-saturation of $\mathcal{A}$. It is immediate that $\mathcal{A}|_{\geq x} = \mathcal{A}'|_{\geq x}$. Moreover, the restriction of $\mathcal{A}$ and $\mathcal{A}'$ to transitions using priorities $\leq x-2$ coincides, so for each $0 \leq y \leq x-2$, relation $\sim_{x-2}$ is a congruence for y-transitions in $\mathcal{A}'$ (and therefore these transitions act uniformly by determinism). The congruence for transitions using priority $>(x-2)$ is preserved as we have only added $(x-1)$-transitions that go to the same $\sim_{x-2}$-class. As $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$, the latter relation is also a congruence in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\mathcal{A}}} = \mathcal{A}'/_{\sim_{\mathcal{A}}}$. ■

LEMMA A.13. *The $(x-1)$-saturation of $\mathcal{A}$ recognises $\mathcal{L}(\mathcal{A})$. Moreover, it is history-deterministic, homogeneous and deterministic over transitions using priorities different from $x-1$.*

PROOF. Let $\mathcal{A}'$ be the $(x-1)$-saturation of $\mathcal{A}$. We have already noted that it is homogeneous and deterministic over transitions using priority different from $x-1$ (Remark A.10). If $w \in \Sigma^\omega$ is accepted by $\mathcal{A}'$ (resp. by $\mathcal{A}$), it is either accepted with an even priority $y < x-1$ or $y > x-1$. In the first case, since $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$, Lemma A.6 allows us to conclude. In the second case, it suffices to apply Lemma A.1.

History-determinism of $\mathcal{A}'$ is clear: one can use a resolver for $\mathcal{A}$. ■

Redundant safe components. From now on, we suppose that $\mathcal{A}$ is $(x-1)$-saturated.

We say that a $(<x)$-safe component S of $\mathcal{A}$ is *redundant* if there is $q \in S$ and $q' \sim_{x-2} q, q' \notin S$, such that $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$. We note that, by normality of $\mathcal{A}$, there are no $(\geq x)$-transitions entering in S; that is, there are no transitions $p \xrightarrow{a:\geq x} q$ with $p \notin S$ and $q \in S$.

REMARK A.14. Automaton $\mathcal{A}$ is $(<x)$-safe centralised if and only if it does not contain any redundant $(<x)$-safe component.

LEMMA A.15. *If $\mathcal{A}$ contains some redundant $(<x)$-safe component, we can find one of them in polynomial time.*

PROOF. The computation of the $(<x)$-safe components of $\mathcal{A}$ can be done by simply a decomposition in SCC of $\mathcal{A}|_{\geq x}$. For each pair of states $q \sim_{x-2} q'$ in different $(<x)$-safe components we just need to check the inclusion $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$, which can be done in polynomial time. ∎

LEMMA A.16. *Let S be a redundant $(<x)$-safe component of $\mathcal{A}$, and let S' be a different $(<x)$-safe component such that there are $q_0 \in S$ and $q'_0 \in S'$, with $q_0 \sim_{x-2} q'_0$ and $\mathsf{Safe}_{<x}(q_0) \subseteq \mathsf{Safe}_{<x}(q'_0)$. Then, for each $q \in S$ there is $q' \in S'$ such that $q \sim_{x-2} q'$ and $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$.*

PROOF. For each $q \in S$, pick $u \in \Sigma^*$ such that $q_0 \xrightarrow{u:\geq x} q$. We let q' be such that $q'_0 \xrightarrow{u} q'$. Since $u \in \mathsf{Safe}_{<x}(q_0) \subseteq \mathsf{Safe}_{<x}(q'_0)$, this latter path produces priority $\geq x$ and, by normality, q' is in S'. As $\sim_{x-2}$ is a congruence for $(\geq x-2)$-transitions, $q' \sim_{x-2} q$. By monotonicity for safe languages (Lemma 5.11), we also have $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(q')$. ∎

Removing redundant safe components. For now on, fix S to be a redundant $(<x)$-safe component of $\mathcal{A}$, and S' a different $(<x)$-safe component as in the previous lemma. For each $q \in S$, we let $f(q) \in S'$ such that $q \sim_{x-2} f(q)$ and $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(f(q))$. We extend f to all of Q by setting it to be the identity over $Q \setminus S$.

We define the automaton $\mathcal{A}'$ as follows:

- The set of states is $Q' = Q \setminus S$.
- The initial state is $f(q_{\mathsf{init}})$.
- For each $p \in Q'$, if $p \xrightarrow{a:y} q$ is a transition in $\mathcal{A}$, we let $p \xrightarrow{a:y} f(q)$ in $\mathcal{A}'$.

We note that, if $q \notin S$, all transitions $p \xrightarrow{a:y} q$ in $\mathcal{A}$ are left unchanged. In particular, the $(\geq x)$-transitions of $\mathcal{A}'$ are the restriction of those appearing in $\mathcal{A}$, $\mathcal{A}'$ is $(x-1)$-saturated, and $(<x)$-transitions in $\mathcal{A}$ not entering in S appear in $\mathcal{A}'$ too. We say that transitions $p \xrightarrow{a:y} q$ of $\mathcal{A}$ such that $q \in S$ have been *redirected in $\mathcal{A}'$*.

LEMMA A.17. *Automaton $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.*

PROOF. We have that $\mathcal{A}'|_{\geq x}$ is the subautomaton of $\mathcal{A}|_{\geq x}$ induced by states in Q'. We show that $\sim_{x-2}$ is $[0, x-2]$-faithful in $\mathcal{A}'$. Transitions that have not been redirected satisfy the congruence requirements, as they satisfy them in $\mathcal{A}$. Let $p \xrightarrow{a:y} q$ and $p' \xrightarrow{a:y'} q'_0$ be two transitions in $\mathcal{A}'$ such that $y \leq x-2$, $p \sim_{x-2} p'$ and such that the second transition has been redirected from $p' \xrightarrow{a:y'} q'$ (the first transition is possibly a redirected one too). By the congruence property in $\mathcal{A}$, we have that $y' = y$ and $q \sim_{x-2} q'$. Since $q' \sim_{x-2} q'_0$, we conclude by transitivity. The equality $\mathcal{A}'/_{\leq x-1}^{\sim_{x-2}} = \mathcal{A}/_{\leq x-1}^{\sim_{x-2}}$ simply follows from the fact that redirected transitions have been defined preserving the $\sim_{x-2}$-classes.

As $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$, the latter relation is also a congruence in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\mathcal{A}}} = \mathcal{A}'/_{\sim_{\mathcal{A}}}$. ∎

LEMMA A.18 (Correctness of the removal of redundant components). *For every state $q' \in Q'$, we have $\mathcal{L}(\mathcal{A}'_{q'}) = \mathcal{L}(\mathcal{A}_{q'})$. In particular, these automata are equivalent. Moreover, automaton $\mathcal{A}'$ is deterministic over transitions with priority different from $x-1$, homogeneous and history-deterministic.*

PROOF. The fact that $\mathcal{A}'$ is deterministic over transitions with priority different from $x-1$ and homogeneous is immediate from its definition. We show the equality of languages for the initial state. The proof is identical for a different state.

The inclusion $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$ directly follows from Lemma A.6.

We describe a sound resolver witnessing $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$ and history-determinism. Take a sound resolver r in $\mathcal{A}$, let $w \in \Sigma^\omega$, and write

$$\rho = p_0 \xrightarrow{w_0} p_1 \xrightarrow{w_1} \dots$$

for the run in $\mathcal{A}$ induced by r over w. We will build a resolver r′ in $\mathcal{A}'$ satisfying the property that the run induced over w, $\rho' = p'_0 \xrightarrow{w_0} p'_1 \xrightarrow{w_1} \dots$ is in one of the following (non-excluding) cases:

a) produces priorities $< x-1$ infinitely often,
b) eventually produces only priorities $\geq x$,
c) $p_i \sim_{x-2} p'_i$ and $\mathsf{Safe}_{<x}(p_i) \subseteq \mathsf{Safe}_{<x}(p'_i)$ for every i sufficiently large.

CLAIM A.19. *A resolver r′ satisfying the property above accepts all words in $\mathcal{L}(\mathcal{A})$.*

Proof. Suppose that $w \in \mathcal{L}(\mathcal{A})$, that is, the run ρ induced by r is accepting. Let ρ' be the run induced over w by r′ in $\mathcal{A}'$. We distinguish two cases, according to the priorities produced by the run ρ in $\mathcal{A}$:

If ρ produces priorities $< x-1$ infinitely often. Then Lemma A.6 allows us to conclude.

If ρ eventually only produces priorities $\geq x-1$. Then, the two first items of Lemma A.6 tells us that ρ' eventually only produces priorities $\geq x-1$ too (so we are not in Case a). We show that ρ' eventually only produces priorities $> x-1$; the last item of Lemma A.6 allows us to conclude (we recall that $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$). If we are in Case b, this property is trivially satisfied. Suppose that we are in Case c, and let $k > 0$ be such that the suffix of ρ from q_k only produces priorities $\geq x$ and such that $\mathsf{Safe}_{<x}(p_i) \subseteq \mathsf{Safe}_{<x}(p'_i)$ for $i \geq k$. Therefore, there is a run over $w_k w_{k+1} \dots$ from p'_k producing exclusively priorities $\geq x$. By determinism over transitions with priority $\geq x$, this run is the one induced by r′. ◆

We finally show how to construct a resolver with this property. We let $p'_0 = f(p_0)$, and assume ρ' constructed up to p'_i satisfying that $p'_i \sim_{x-2} p_i$.

If there is a transition $p'_i \xrightarrow{w_i:y} p'_{i+1}$ with $y \neq x-1$, then we take this one (there is no other option), which satisfies $p_{i+1} \sim_{x-2} p'_{i+1}$ by Lemma A.17. Moreover, if $y \geq x$ and $\mathsf{Safe}_{<x}(p_i) \subseteq$

$\mathsf{Safe}_{<x}(p'_i)$, then $\mathsf{Safe}_{<x}(p_{i+1}) \subseteq \mathsf{Safe}_{<x}(p'_{i+1})$ by Lemma 5.11. If there is a transition $p'_i \xrightarrow{w_i:x-1}$, we take $p'_{i+1} = f(p_{i+1})$ (this transition exists in $\mathcal{A}'$ by $x-1$-saturation, as $p'_i \sim_{x-2} p_i$).

We show that this resolver satisfies the desired property. Suppose that we are not in the two first cases, that is, ρ' eventually only produces priorities $\geq x-1$, and it produces priority $x-1$ infinitely often. Take a suffix $p'_k \xrightarrow{w_k:y_k} p'_{k+1} \xrightarrow{w_{k+1}:y_{k+1}} \dots$ of ρ' such that no priority $< x-1$ is produced and such that $y_k = x-1$. Then, by definition of the transitions using $x-1$ chosen by the resolver, $p'_{k+1} = f(p_{k+1})$, so $\mathsf{Safe}_{<x}(p_{k+1}) \subseteq \mathsf{Safe}_{<x}(p'_{k+1})$. We conclude by induction, as transitions taken by the resolver using priorities $\geq x-1$ preserve the inclusion of $(<x)$-safe languages. ∎

Transformation preserves being a structured signature automaton. To be able to finish the proof of Lemma 5.16, we just need to show that $\mathcal{A}'$ is a $(x-2)$-structured signature automaton. We give some technical lemmas that will help us show this.

LEMMA A.20. *Let q and p be two states of $\mathcal{A}'$. There is a path $q \xrightarrow{w:x-1} p$ in $\mathcal{A}$ if and only if there is a path $q \xrightarrow{w:x-1} p$ in $\mathcal{A}'$.*

PROOF. If a path $q \xrightarrow{w:x-1} p$ appears in $\mathcal{A}'$, the very same path also exists in $\mathcal{A}$.

Suppose now that a path $\rho = q \xrightarrow{w:x-1} p$ exists in $\mathcal{A}$. Let S be the $<x$-safe component that has been removed from $\mathcal{A}$. If the path ρ does not cross S, then it also appears in $\mathcal{A}$. Suppose that it enters in S. We remark that, by normality and by the definition of safe component, each time that ρ enters or exists S, it produces priority $x-1$. We consider the last time that ρ enters and exists S:

$$q \xrightarrow{u_1:\geq x-1} q_1 \xrightarrow{u_2:\geq x} q_2 \xrightarrow{a:x-1} q_3 \xrightarrow{u_3:\geq x-1} p,$$

with $w = u_1u_2au_3$, $q_3 \notin S$, and the path $q_3 \xrightarrow{u_3} p$ does not enter S (so it also appears in $\mathcal{A}'$). Consider any run over u_1u_2 from q in $\mathcal{A}'$:

$$q \xrightarrow{u_1:\geq x-1} q'_1 \xrightarrow{u_2:\geq x-1} q'_2.$$

As $\sim_{x-2}$ is a $[0, x-2]$-faithful congruence, we have that $q_2 \sim_{x-2} q'_2$. As $\mathcal{A}'$ is $x-1$-saturated, there is a transition $q'_2 \xrightarrow{a:x-1} q_3$. Therefore, we obtain in $\mathcal{A}'$ the path:

$$q \xrightarrow{u_1:\geq x-1} q'_1 \xrightarrow{u_2:\geq x-1} q'_2 \xrightarrow{a:x-1} q_3 \xrightarrow{u_3:\geq x-1} p.$$ ∎

LEMMA A.21. *Let q and p be two states of $\mathcal{A}'$ and let y be any priority. There is a path $q \xrightarrow{w:y} p$ in $\mathcal{A}$ if and only if there is a path $q \xrightarrow{w':y} p$ in $\mathcal{A}'$.*

PROOF. If $y \geq x$, q and p are in the same $(<x-1)$-safe component. Since the $(<x-1)$-safe components in $\mathcal{A}'$ are safe components in $\mathcal{A}$, the result is clear in this case.

If $y = x-1$, the result is assured by the previous Lemma A.20.

Assume $y < x-1$. Suppose that there is a path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ (the proof is analogous if we take this path in $\mathcal{A}'$), and let $q \overset{w:y'}{\rightsquigarrow} p'$ be the run over w from q in $\mathcal{A}'$. As $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation, by Lemma A.4, we have that $y' = y$ and $p \sim_{x-2} p'$. As $\mathcal{A}$ satisfies Item 5 from the definition of a structured signature automaton, there is a path $p' \overset{u:\geq x-1}{\rightsquigarrow} p$ in $\mathcal{A}$. By Lemma A.20, such a path also exists in $\mathcal{A}'$, so we can take $w' = wu$, giving us a path $q \overset{w:y}{\rightsquigarrow} p' \overset{u:>y}{\rightsquigarrow} p$. ■

Previous lemma tells us, in particular, that for every $(<)y$-safe component $S_i^{<y}$ of $\mathcal{A}$, the intersection of $S_i^{<y}$ with Q' constitute the states of a $(<)y$-safe component in $\mathcal{A}'$. Therefore, automaton $\mathcal{A}'$ inherits the decomposition in $(<)y$-safe component $S_1^{<y}, \dots, S_{k_y}^{<y}$ from $\mathcal{A}$, for each y; we simply remove those components whose intersection with Q' is empty.

LEMMA A.22. *Automaton $\mathcal{A}'$ is a $(x-2)$-structured signature automaton.*

PROOF. We go through all the conditions of the definition of a $(x-2)$-structured signature automaton. We recall that, by Lemma A.17, the relation $\sim_{x-2}$ is $[0, x-2]$-faithful congruence in $\mathcal{A}'$.

Normal form. We check that $\mathcal{A}'$ satisfies the hypothesis of the characterisation from Lemma A.8. We let $q' \xrightarrow{a:y} p'$ be a transition in $\mathcal{A}'$. If it is not a redirected transition, it exists in $\mathcal{A}$, so we can conclude by normalisation of $\mathcal{A}$ and Lemma A.21. Assume that $q' \xrightarrow{a:y} p'$ is a transition that has been redirected from $q' \xrightarrow{a:y} p$. In particular, $p \sim_{x-2} p'$ and $y < x$. By Item 5 of the definition of structured signature automaton applied to $\mathcal{A}$, there is a path $p' \overset{w:>y}{\rightsquigarrow} p$ in $\mathcal{A}$, and by normality, there is a returning paths $p \overset{u_1:y}{\rightsquigarrow} q$ and $p \overset{u_2:y-1}{\rightsquigarrow} q$. Again, Lemma A.21 allows us to find the desired returning paths in $\mathcal{A}'$.
Item 1. By Lemma A.18, the residuals in $\mathcal{A}'$ correspond to those in $\mathcal{A}$, so preorder $\sqsubseteq_0$ correspond to their inclusion in $\mathcal{A}'$ too.
Item 2. By the previous remarks, for all y, the $(<y)$-safe components of $\mathcal{A}'$ are obtaining by taking the intersection with those in $\mathcal{A}$. Therefore, odd preorders $\sqsubseteq_{y-1}$ correspond to the order of $(<y)$-safe components on $\mathcal{A}'$.
Item 3. As $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$, for every $y < x-1$ and state q' in $\mathcal{A}'$, $\mathsf{Safe}^{\mathcal{A}'}_{<y}(q') = \mathsf{Safe}^{\mathcal{A}}_{<y}(q')$. Therefore, the preorders at even levels $\sqsubseteq_y$ correspond to the inclusion of safe languages in $\mathcal{A}'$, as they do in $\mathcal{A}$.
Item 4. Let q and q' be two states in $\mathcal{A}'$ such that $q \sim_y q'$, for $y \leq x-2$ even, and let $q \xrightarrow{a:y'} p$ for $y' \leq y$ and $q' \xrightarrow{a:z} p'$. As $\mathcal{A}'$ is a nice transformation, $y' = z$. If the first of these transitions is not a redirected one, then, by strong congruence of $\leq(x-2)$-priorities in $\mathcal{A}$, neither is the second one, and $p = p'$. Assume that these transitions have been redirected from, $q \xrightarrow{a:y'} p_1$ and $q' \xrightarrow{a:y'} p_1'$. Then, as Item 4 is satisfied in $\mathcal{A}$, $p_1 = p_1'$, so $p = f(p_1) = f(p_1') = p'$.
Item 5. Directly follows from Lemma A.21 and the fact that $\mathcal{A}$ satisfies this property.
Item 6. Follows from the fact that if $q \not\rightsquigarrow_{y-1} p$ in $\mathcal{A}'$, then $q \not\rightsquigarrow_{y-1} p$ in $\mathcal{A}$; and the equality $\mathsf{Safe}^{\mathcal{A}'}_{<y}(q) = \mathsf{Safe}^{\mathcal{A}}_{<y}(q)$. ■

Obtaining Lemma 5.16. We have all the necessary elements to deduce Lemma 5.16. Using Lemma A.15, we can decide whether $\mathcal{A}$ contains a redundant $(<x)$-safe component in polynomial time. If it contains none, $\mathcal{A}$ is already $(<x)$-safe centralised. While we can find redundant safe components, we remove them applying the transformation described above. This transformation can clearly be done in polynomial time, and by Lemmas A.18 and A.22, the obtained automaton recognises the correct language and preserves all the hypothesis assumed in the induction.

A.2.3 Existence of uniform words and synchronising separating runs

We now provide proofs for Lemmas 5.18 and 5.19. In Section 5.2, we derived totality of the order $\sqsubseteq_x$ in each $\sim_{x-1}$-component from these lemmas (c.f. Lemma 5.20).

Hypothesis. In all the subsection we assume that x is an even priority and $\mathcal{A}$ is a $(x-2)$-structured signature automaton with initial state q_{init} that is moreover:

- deterministic over transitions with priority different from $x-1$,
- homogeneous,
- history-deterministic, and
- $(<x)$-safe centralised.

Words producing priority x uniformly.

LEMMA 5.18 (Existence of uniform words). (Restated) *Let p and q be two states from the same $(<x)$-safe component. There is a word $w \in \Sigma^*$ producing priority x uniformly in $[q]_x$ leading to $[p]_x$.*

PROOF. The fact that for such paths we must have $p_1 \sim_x p_2$ follows from the monotonicity of safe languages (Lemma 5.11) and the fact that $(\geq x)$-transitions preserve $\sim_{x-1}$-classes.

We let $\{q_1, q_2, \ldots, q_k\}$ be an enumeration of the states of $[q]_x$.

CLAIM A.23. *For each $q_i \in [q]_x$ there is a word $u_i \in \Sigma^*$ such that $q_i \xrightarrow{u_i:x} q_i$.*

Proof. Since q_i and q are in the same $(<x)$-safe component, there is a word u_i' such that $q_i \xrightarrow{u_i':\geq x} q$. By normality, there is a word u_i'' such that $q \xrightarrow{u_i'':x} q_i$. We just take $u_i = u_i' u_i''$. ◆

We will define k finite words $w_1, w_2 \ldots, w_k \in \Sigma^*$ satisfying:

- For $q' \in [q]_x$ and for every $i \leq k$, $q' \xrightarrow{w_1 w_2 \ldots w_i:\geq x} [q]_x$.
- $q_j \xrightarrow{w_1 w_2 \ldots w_i:x} [q]_x$ for every $j \leq i$.

In order to obtain these properties, we just define recursively $w_1 = u_1$ and $w_i = u_j$, for u_j as given by the previous claim, if:

$$q_i \xrightarrow{w_1 w_2 \ldots w_{i-1}:\geq x} q_j.$$

Finally, we let $w = w_1 w_2 \ldots w_k w$, which first produces priority x when read from any state of $[q]_x$, and then goes to the $\sim_x$-class of p producing priorities $\geq x$. ∎

Resolvers implemented by finite memories. For the upcoming proofs, we need to introduced the notion of memories for resolvers. Let $\mathcal{A} = (Q, \Sigma, q_{\mathsf{init}}, \Delta, \mathsf{p})$ be a non-deterministic automaton. A *memory structure* for $\mathcal{A}$ is a tuple $(M, m_{\mathsf{init}}, \mu, \sigma)$, where M is a set of memory states, $m_{\mathsf{init}} \in M$ is an initial state, $\mu : M \times \Delta \to M$ is an update function and $\sigma \colon Q \times M \times \Sigma \to \Delta$. It *implements* a resolver r if for all $a \in \Sigma$, $\mathsf{r}(\varepsilon, a) = \sigma(q_{\mathsf{init}}, m_{\mathsf{init}}, a)$ and for all $\rho \in \Delta^+$ ending in p, $\mathsf{r}(\rho, a) = \sigma(p, \mu(m_{\mathsf{init}}, \rho), a)$.

LEMMA A.24 ([6]). *Every history-deterministic ε-completion admits a sound resolver implemented by a finite memory structure.*

In the rest of the subsection we fix a sound resolver r for $\mathcal{A}$ implemented by a memory structure $\mathcal{M} = (M, m_{\mathsf{init}}, \mu)$. For simplicity, we assume that every pair of a state and a memory state (q, m) is *reachable using* r, that is, there is some word $w \in \Sigma^*$ such that $\rho = q_{\mathsf{init}} \overset{w}{\underset{\mathsf{r}}{\rightsquigarrow}} q$ and $\mu(m_{\mathsf{init}}, \rho) = m$. It is easy to see that we can get rid of this assumption in the upcoming proof just by ignoring pairs (q, m) that are not reachable.

For $(q, m) \in Q \times M$, we let $(q, m) \overset{w}{\underset{\mathsf{r}}{\rightsquigarrow}} (q', m')$ be the (unique) run induced by r from q when the memory structure is in state m. We extend notations of the form $(q, m) \overset{w:x}{\underset{\exists, \mathsf{r}}{\rightsquigarrow}} q'$ in the natural way; the previous one means that there exists $u_0 \in \Sigma^*$ such that the induced run of r is $\rho = q_{\mathsf{init}} \overset{u_0}{\underset{\mathsf{r}}{\rightsquigarrow}} q$, $\mu(m_{\mathsf{init}}, \rho) = m$ and $q_{\mathsf{init}} \overset{u_0 w}{\underset{\mathsf{r}}{\rightsquigarrow}} q'$, producing priority x in the second part of this run.

As $\mathcal{A}$ is deterministic over transitions using priorities $\geq x$, we may omit the subscript r in paths producing no priority $< x$.

Synchronising separating runs.

LEMMA 5.19 (Synchronisation of separating runs). (Restated) *Suppose that $q \sim_{x-1} q'$ and $q \not\sim_x q'$ and let $p \in [q]_{x-1}$. There is a word $w \in \Sigma^+$ such that $[q]_x \overset{w:x-1}{\underset{\forall, \mathsf{r}}{\rightsquigarrow}} [p]_x$ and $[q']_x \overset{w:x}{\underset{\forall, \mathsf{r}}{\rightsquigarrow}} [p]_x$.*

PROOF. We first show that we can force to produce priority $x - 1$ from $[q]_x$, while remaining safe from $[q]_x$.

CLAIM A.25. *There is a word $u \in \Sigma^+$ such that for all $s \in [q]_x$:*

$$s \overset{u:x-1}{\underset{\forall, \mathsf{r}}{\rightsquigarrow}}, \quad \textit{and} \quad [q']_x \overset{u:x}{\underset{\forall, \mathsf{r}}{\rightsquigarrow}} [p]_x.$$

Proof. By definition of the preorder $\sqsubseteq_x$, there is a word $u_1 \in \Sigma^+$ such that for all $s \in [q]_x$ and $s' \in [q']_x$, $s \xrightarrow[\forall,\mathsf{r}]{u_1:x-1}$ and $s' \xrightarrow[\mathsf{r}]{u_1:\geq x}$. By normality, we can extend this word so that $q' \xrightarrow[\mathsf{r}]{u_1':\geq x} q'$; by monotonicity of safe languages all runs from $[q']_x$ reading u_1' go back to $[q']_x$. Applying Lemma 5.18, we obtain u_2 that produces priority x uniformly in $[q']_x$ and goes to $[p]_x$. We take $u = u_1'u_2$, which satisfies $[q']_x \xrightarrow[\mathsf{r}]{u:x} [p]_x$, and it produces at least one occurrence of priority $x-1$ from every state $s \in [q]_x$. By $[0, x-2]$-faithfulness, a run $s \xrightarrow{u}$ only produces priorities $\geq x-1$, which concludes. ◆

CLAIM A.26. *Let $p' \sim_{x-2} q$ and let $m \in M$ be a memory state. Then there is a word $w_{q,m}$ such that:*

$$(q,m) \xrightarrow[\mathsf{r}]{w_{q,m}:\geq x-1} [p']_x, \ \ \textit{and} \quad [p']_x \xrightarrow[\forall,\mathsf{r}]{w_{q,m}:x} [p']_x.$$

Proof. We distinguish two cases. First, assume that $\mathsf{Safe}_{<x}(q) \subseteq \mathsf{Safe}_{<x}(p')$. In this case, by $<x$-safe centrality of $\mathcal{A}$, q and p' are in the same $<x$-safe component so $q \sim_{x-1} p'$ (and $q \sqsubseteq_x p'$). Let $p'_{\max}$ be a state in $[p']_{x-1}$ such that $p' \sqsubseteq_x p'_{\max}$, and maximal with this property. Let $w_1 \in \Sigma^*$ be a word such that $q \xrightarrow{w_1:\geq x} p'_{\max}$ (which exists because these two states are in the same $<x$-safe component). By monotonicity of safe languages, $p' \xrightarrow{w_1:\geq x} p''$ with $p'_{\max} \sqsubseteq_x p''$. By maximality of $p'_{\max}$, we must have $p'' \sim_x p'_{\max}$. Finally, let $w_2 \in \Sigma^*$ such that $p'' \xrightarrow{w_2:x} p'$ producing priority x uniformly in the class $[p'']_x$ (which exists by Lemma 5.18). We obtain $q \xrightarrow{w_1:\geq x} p'_{\max} \xrightarrow{w_2:x} [p']_x$ and $[p']_x \xrightarrow{w_1:\geq x} [p'_{\max}]_x \xrightarrow{w_2:x} [p']_x$ as required.

Assume now that $\mathsf{Safe}_{<x}(q) \not\subseteq \mathsf{Safe}_{<x}(p')$. In that case, we can find a word $w_1 \in \mathsf{Safe}_{<x}(p') \setminus \mathsf{Safe}_{<x}(q)$. By Lemma 5.18, we may assume that it produces priority x uniformly from $[p']_x$ and comes back to this class. Moreover, by faithfulness, it cannot produce priorities $\leq x-2$ and respects the $\sim_{x-2}$-classes. Thus:

$$(q,m) \xrightarrow[\mathsf{r}]{w_1:x-1} (q_1,m_1), \ \text{with } q_1 \sim_{x-2} p', \text{ and } \quad [p']_x \xrightarrow{w_1:x} [p']_x.$$

If $\mathsf{Safe}_{<x}(q_1) \subseteq \mathsf{Safe}_{<x}(p')$, we can conclude by using the first case. While we do not have this inclusion, we build, using the argument above, a sequence of words $w_1, w_2, \dots$ such that:

$$(q_i,m_i) \xrightarrow[\mathsf{r}]{w_i:x-1} (q_{i+1},m_{i+1}), \text{ and } [p']_x \xrightarrow{w_i:x} [p']_x.$$

This sequence cannot be infinite. If it were the case, resolver r would induce a rejecting run over $w_1w_2\dots$ from (q,m), and an accepting from p'. This is a contradiction, as the equivalence $q \sim_{x-2} p$ implies $\mathcal{L}(\mathcal{A}_q) = \mathcal{L}(\mathcal{A}_{p'})$ (since $\sim_{x-2}$ refines $\sim_{\mathcal{A}}$). Therefore, for some k we must have $\mathsf{Safe}_{<x}(q_k) \subseteq \mathsf{Safe}_{<x}(p')$ and we can extend the path as wanted using the first case. ◆

We may finally deduce the result of Lemma 5.19 from the two previous claims. First, we read the word u from Claim A.25, which forces to produce priority $x-1$ from any state in $[q]_x$. We now show how to use Claim A.26 to redirect each state, one by one, to the class $[p]_x$.

We let $(q_1, m_1), \dots, (q_k, m_k)$ be an enumeration of all states such that there exists $s \in [q]_x$ with $s \xrightarrow[\exists, r]{u:x-1} (q_i, m_i)$. We note that, by $[0, x-2]$-faithfulness, $q_i \sim_{x-2} q \sim_{x-2} p$. We recursively build a sequence of k words, $w_1, \dots, w_k \in \Sigma^*$ by setting:

$$(q_i, m_i) \xrightarrow[r]{w_1 \dots w_{i-1}:\geq x-1} (q_i', m_i') \xrightarrow[r]{w_i:\geq x-1} [p]_x \quad \text{and} \quad [p]_x \xrightarrow[\forall, r]{w_i:x} [p]_x.$$

We can indeed do this by letting $w_i = w_{q_i', m_i'}$ as given by Claim A.26, as by $[0, x-2]$-faithfulness $q_i' \sim_{x-2} p$.

The word $w_1 \dots w_i$ satisfies that, for $j \leq i$, $(q_j, m_j) \xrightarrow[r]{w_1 \dots w_i:\geq x-1} [p]_x$. We conclude the proof of the lemma by putting $w' = uw_1 \dots w_k$. ∎

A.2.4 Re-determinisation

We give the proof of Lemma 5.21, that is, we show that we can obtain an equivalent deterministic automaton from $\mathcal{A}$ while preserving all the obtained structure of total preorders satisfying the conditions of a structured signature automaton.

Hypothesis. We assume that $\mathcal{A}$ is a parity automaton recognising W with nested total preorders defined up to $\sqsubseteq_x$ such that:

- it is a $(x-2)$-structured signature automaton,
- preorder $\sqsubseteq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton,
- preorder $\sqsubseteq_x$ satisfies the property from Item 3 from the definition of a structured signature automaton,
- it is deterministic over transitions with priorities different from $x-1$,
- it is homogeneous, and
- it is history-deterministic.

Obtaining a deterministic automaton.

LEMMA 5.21 (Re-determinisation). (Restated) *There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ with nested total preorders defined up to $\sqsubseteq_x$ satisfying that:*

- *it is a $(x-2)$-structured signature automaton,*
- *preorder $\sqsubseteq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and*
- *preorder $\sqsubseteq_x$ is a congruence and satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4 .*

Moreover, automaton $\mathcal{A}'$ can be computed in polynomial time from $\mathcal{A}$ and $|\mathcal{A}'| \leq |\mathcal{A}|$.

An intuitive idea for the construction of $\mathcal{A}'$ was given in Section 5.2. We formalise it and prove its correctness now.

Automaton $\mathcal{A}'$ is obtained by keeping all the structure of $\mathcal{A}$, except for $(x-1)$-transitions; for each state q and letter a such that some transition $q \xrightarrow{a:x-1} p$ appears in $\mathcal{A}$, we will redefine this transition as $q \xrightarrow{a:x-1} p'$ for some $p' \sim_{x-2} p$ as determined next.

For each $\sim_{x-1}$-class $[q]_{x-1}$ of $\mathcal{A}$, we pick a state in the class that is maximal for $\sqsubseteq_x$. We let $f(q)$ be that state. That is, for two states $q_1 \sim_{x-1} q_2$:

— $f(q_1) = f(q_2)$,

— $f(q_1) \in [q_1]_{x-1}$, and

— $q_1 \sqsubseteq_x f(q_1)$.

Recall the total order over the $<x$-safe components of $\mathcal{A}$ given by $S_1^{<x}, S_2^{<x}, \dots, S_{k_x}^{<x}$. Let $q \in S_i^{<x}$ and $a \in \Sigma$ such that $q \xrightarrow{a:x-1} p$ appears in $\mathcal{A}$. If it exists, we let i_{next} be the maximal $0 \leq i_{\mathsf{next}} < i$ such that there is some $p' \in [p]_{x-2}$, $p' \in S_{i_{\mathsf{next}}}^{<x}$. If index i_{next} is not defined, we let it be the maximal index $i \leq i_{\mathsf{next}} \leq k_x$ with the previous property. We fix a state $p_{q,a} \in S_{i_{\mathsf{next}}}^{<x}$ with $p_{q,a} \in [p]_{x-2}$. We let the a-transition from q in $\mathcal{A}'$ be $q \xrightarrow{a:x-1} f(p_{q,a})$. This completes the description of $\mathcal{A}'$. It is indeed deterministic, as $\mathcal{A}$ is homogeneous and deterministic over transitions with priorities different from $x-1$.

We can find a maximal state $f(q)$ for $\sqsubseteq_x$ in the class $[q]_{x-1}$ in polynomial time, as the comparison of $<x$-safe languages can be done in polynomial time (Lemma 6.1). Therefore, automaton $\mathcal{A}'$ can be built in polynomial time.

LEMMA A.27. *Automaton $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.*

PROOF. This is clear, as the restriction of $\mathcal{A}'$ to transitions using a priority different from $(x-1)$ coincides with that of $\mathcal{A}$, and every transition $q \xrightarrow{a:x-1} p'$ in $\mathcal{A}'$ comes from a transition $q \xrightarrow{a:x-1} p$ in $\mathcal{A}$ with $p \sim_{x-2} p'$. ■

LEMMA A.28. *For every state $q \in Q$, we have $\mathcal{L}(\mathcal{A}'_q) = \mathcal{L}(\mathcal{A}_q)$. In particular, automaton $\mathcal{A}'$ recognises the language $\mathcal{L}(\mathcal{A})$.*

PROOF. For simplicity, we give the proof just for the initial state; the proof being identical for any other state.

Let $w \in \Sigma^\omega$. If the minimal priority produced infinitely often by the run over w in $\mathcal{A}'$ is $y < x-1$ or $y > x-1$, then w is accepted by $\mathcal{A}'$ if and only if w is accepted by $\mathcal{A}$, by Lemma A.6 and the fact that $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation of $\mathcal{A}$ at level $x-1$.

Assume that the minimal priority produced infinitely often by the run over w in $\mathcal{A}'$ is $x-1$ (so it is rejecting), and suppose by contradiction that w is accepted by $\mathcal{A}$. By Lemma A.6, an accepting run over w in $\mathcal{A}$ cannot produce a priority $y < x$ infinitely often. Therefore, it eventually remains in a $<x$-safe component $S_{i_{\mathcal{A}}}^{<x}$. Let ρ be such an accepting run, and let ρ' be

the run over w in $\mathcal{A}'$. We represent them as:

$$\rho = q_0 \overset{u}{\rightsquigarrow} q_N \xrightarrow{w_N:\geq x} q_{N+1} \xrightarrow{w_{N+1}:\geq x} \dots, \quad \rho' = q'_0 \overset{u}{\rightsquigarrow} q'_N \overset{w_N:x'_N}{\rightsquigarrow} q'_{N+1} \xrightarrow{w_{N+1}:x'_{N+1}} \dots,$$

where u is the prefix of size N of w, $q_0 = q'_0 = q_{\mathsf{init}}$, and we suppose that $q_k \in S^{<x}_{i_{\mathcal{A}}}$ for all $k \geq N$. As $\mathcal{A}'$ is a $\sim_{x-2}$-nice transformation at level $x-1$, we have that $q_k \sim_{x-2} q'_k$ for all k. Let $k_1 < k_2 < k_3 \dots$ be the positions greater than N where $x'_{k_j} = x-1$, and let $i_1, i_2, \dots$ be the indices of the $<x$-safe components such that $q_{k_j+1} \in S^{<x}_{i_j}$, that is, when taking the transition $q_{k_j} \xrightarrow{w_{k_j}:x-1}$ we land in $S^{<x}_{i_j}$.

CLAIM A.29. *Eventually, $i_j = i_{\mathcal{A}}$.*

Proof. Consider a transition $q'_{k_j} \xrightarrow{w_{k_j}:x-1} q'_{k_j+1}$, and suppose first that $i_{\mathcal{A}} < i_{j-1}$. We claim that $i_{\mathcal{A}} \leq i_j < i_{j-1}$. This would end the proof, as we obtain a strictly decreasing sequence of indices bounded by $i_{\mathcal{A}}$. In order to determine i_j, we need to look at the definition of i_{next}. As $q_k \sim_{x-2} q'_k$ for all k, there are always states in $[q'_{k_j+1}]_{x-2}$ in some $<x$-safe components with an index $i_{\mathcal{A}} \leq i < i_{j-1}$. Thus, we obtain the desired result by definition of i_{next}.

If $i_{j-1} \leq i_{\mathcal{A}}$, by definition of i_{next}, there is be a sequence of decreasing indices $i_{j-1} > i_j > i_{j+1} > \dots$ until no $\sim_{x-2}$-equivalent state appears in a strictly smaller safe component. By the same argument as before, there is always a $\sim_{x-2}$-equivalent state in $S^{<x}_{i_{\mathcal{A}}}$, so eventually $i_{\mathcal{A}} \leq i_{j'}$, and either this is an equality, or we reduce to the previous case. ◆

Let j be the first position such that $i_j = i_{\mathcal{A}}$, and consider transitions $q_{k_j} \xrightarrow{w_{k_j}:\geq x} q_{k_j+1}$ and $q'_{k_j} \xrightarrow{w_{k_j}:x-1} q'_{k_j+1}$ in ρ and ρ', respectively. By definition of $x-1$-transitions of $\mathcal{A}'$, the state we go to in ρ' is $q'_{k_j+1} = f(q_{k_j+1})$. As we have chosen $f(q_{k_j+1})$ maximal in its $\sim_{x-1}$-class, we have $q_{k_j+1} \sqsubseteq_x q'_{k_j+1}$, so we have the inclusion of $<x$-safe languages between these states. Therefore, if there is a $<x$-safe run over w' from q_{k_j+1} in $\mathcal{A}$, there is also such a safe run over w' from q'_{k_j+1} in $\mathcal{A}'$. This contradicts the fact that the run ρ' produces priority $x-1$ infinitely often, while the run ρ is $<x$-safe from q_{k_j+1}, concluding the proof. ■

To finish the proof of Lemma 5.21, we just need to show that $\mathcal{A}'$ preserves all the properties of the preorders induced by $\mathcal{A}$. As in the previous section, to obtain normality of the automaton and Item 5, we rely on a technical lemma that tells us that we can connect states in the same $\sim_{x-2}$-component as desired.

LEMMA A.30. *Let $q \sim_{x-2} p$ be two different states in Q. There is a word $w \in \Sigma^*$ and a path $q \overset{w:>x-2}{\rightsquigarrow} p$ in $\mathcal{A}'$*

PROOF. Let i_p be the index such that $p \in S^{<x}_{i_p}$. If q belongs to this same safe component, we can connect both states by a path producing priorities $\geq x$. If not, by $<x$-safe centrality of $\mathcal{A}$,

there is a word $w_1 \in \mathsf{Safe}_{<x}(p) \setminus \mathsf{Safe}_{<x}(q)$. We let $q \xrightarrow{w_1:x-1} q_1$ and $p \xrightarrow{w_1:\geq x} p_1$. We have that $q_1 \sim_{x-2} p_1$ and $p_1 \in S^{<x}_{i_p}$. While $q_j \notin S^{<x}_{i_p}$, we extend this run in a similar way. Using the same argument as in the proof of the previous lemma, by definition of i_{next} each time that the run from q sees a priority $x-1$ it decreases the index of its safe component, so eventually it must land in $S^{<x}_{i_p}$. ■

LEMMA A.31. *Let $q \xrightarrow{w:y} p$ be a path in $\mathcal{A}$. There is a path $q \xrightarrow{w':y} p$ in $\mathcal{A}'$ connecting the same states and producing the same minimal priority.*

PROOF. If $y \geq x$ we can just take $w = w'$. Suppose $y < x$ and consider the run $q \xrightarrow{w:y'} p'$ in $\mathcal{A}'$. By Lemma A.4, $p \sim_{x-2} p'$. Also, $y = y'$ (if $y < x-1$, this is given by Lemma A.4, if $y = x-1$, by the definition of $\mathcal{A}'$). By the previous lemma, we can extend this run to $p' \xrightarrow{w_2:\geq x-2} p$, and take $w' = ww_2$. ■

LEMMA A.32. *Automaton $\mathcal{A}'$, with the preorders $\sqsubseteq_0, \dots, \sqsubseteq_x$ inherited from $\mathcal{A}$ satisfies:*

- *it is a $(x-2)$-structured signature automaton,*
- *preorder $\sqsubseteq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and*
- *preorder $\sqsubseteq_x$ satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4.*

PROOF. We start verifying the properties for the preorders $\sqsubseteq_{x-1}$ and $\sqsubseteq_x$. We note that the $<x$-safe components of $\mathcal{A}'$ exactly correspond to those in $\mathcal{A}$, and that for every $q \in Q$, $\mathsf{Safe}^{\mathcal{A}}_{<x}(q) = \mathsf{Safe}^{\mathcal{A}'}_{<x}(q)$. The fact that $\sqsubseteq_{x-1}$ satisfies Items 2 and 6, and that $\sqsubseteq_x$ satisfies Item 3 follows immediately. We check that relation $\sim_x$ satisfies Item 4 for priorities $y < x$. For $y \leq x-2$, this follows from the fact that $\sim_x$ refines $\sim_{x-2}$, and the latter relation satisfies Item 4. For $y = x-1$, if $q \xrightarrow{a:x-1} p$ and $q' \xrightarrow{a:x-1} p'$, with $q \sim_x q'$, by definition of the $x-1$-transitions in $\mathcal{A}'$, $p = f(p) = f(p') = p'$.

Checking that $\mathcal{A}'$ is a $(x-2)$-structured signature automaton poses no difficulty. It can be done in an analogous way as it was done in the proof of Lemma A.22; by applying Lemma A.31 to obtain normality of $\mathcal{A}'$ and property from Item 5. ■

A.2.5 Uniformity of x-transitions over $\sim_x$-classes

We finally show how to transform $\mathcal{A}$ into an equivalent automaton that is either x-structured signature, or strictly smaller. The techniques presented here generalise those applying to Büchi automata appearing in Section 4.3 of the warm-up.

Hypothesis. In all this subsection we suppose that $\mathcal{A}$ is deterministic parity automaton recognising W with nested total preorders defined up to $\sqsubseteq_x$ such that:

- it is a $(x-2)$-structured signature automaton,
- preorder $\sqsubseteq_{x-1}$ satisfies properties from Items 2 and 6 from the definition of a structured signature automaton, and
- preorder $\sqsubseteq_x$ is a congruence and satisfies the property from Item 3 and, for priorities $y < x$, also that from Item 4 .

Our objective is to prove, under this list of hypothesis, that we can either obtain an equivalent deterministic x-structured signature automaton, or reduce the number of states of $\mathcal{A}$.

LEMMA 5.22 (Uniformity of x-transitions over $\sim_x$-classes). (Restated) *There is a deterministic parity automaton $\mathcal{A}'$ equivalent to $\mathcal{A}$ such that either:*

- *$\mathcal{A}'$ is an x-structured signature automaton with $|\mathcal{A}'| \leq |\mathcal{A}|$, or*
- *$|\mathcal{A}'| < |\mathcal{A}|$.*

In both cases, such an automaton can be computed in polynomial time from $\mathcal{A}$.

We remark that $\sim_x$ already satisfies most desired properties of monotonicity; only the uniformity for x-transitions is missing.

LEMMA A.33. *The relation $\sim_x$ is a $[0, x-1]$-faithful congruence. Moreover, over each $\sim_{x-1}$-class, transitions using priorities $\geq x$ are monotone for $\sqsubseteq_x$.*

PROOF. The $[0, x-2]$-faithfulness follows from the fact that $\sim_x$ refines $\sim_{x-2}$ and $\mathcal{A}$ is $(x-2)$-structured signature. The uniformity of $(x-1)$-transitions over $\sim_x$-classes is given by the fact that $\sim_x$-equivalent states have the same $<x$-safe language, combined with the uniformity of $<(x-1)$-transitions.

The fact that $\sim_x$ is a congruence for $(x-1)$-transitions follows from Item 4 of the definition of a structured signature automaton (we recall that $\sim_x$ satisfies this property by hypothesis). The congruence for $\geq x$-transitions and the monotonicity of $\geq x$-transitions for $\sqsubseteq_x$ at each $\sim_{x-1}$-class follow from the monotonicity of $<x$-safe languages (Lemma 5.11). ■

Polished automata. We generalise the notion of polished automata from Section 4.3 to our current setting. Recall that $[q]_x$ is the class of q for the equivalence relation $\sim_x$.

DEFINITION A.34 (Polished classes and automata). We say that the class $[q]_x$ is *x-polished* if:

- Words producing priority x act uniformly in $[q]_x$. That is, if $q_1, q_2 \in [q]_x$ and $q_1 \xrightarrow{w:x}$, then $q_2 \xrightarrow{w:x}$.
- For every $q_1, q_2 \in [q]_x$, $q_1 \neq q_2$, there is a path $q_1 \xrightarrow{w:>x} q_2$ producing exclusively priorities $> x$ joining q_1 and q_2.

We say that the automaton $\mathcal{A}$ is *x-polished* if all its $\sim_x$-classes are x-polished.

REMARK A.35. We remark that, as $x \geq 2$ and we assume that the automaton $\mathcal{A}$ is in normal form, all non-trivial $\sim_x$-classes are recurrent: if a $\sim_x$-class is not trivial, there is a cycle visiting all the states of the class. Therefore, we do not need to take care of transient classes (as it was the case in Lemma 4.21 from the warm-up).

LEMMA A.36. *We can decide whether $\mathcal{A}$ is x-polished in polynomial time.*

PROOF. As $\sim_x$ is a $[0, x-1]$-faithful congruence (Lemma A.33), we just need to check the first property for letters, which can be done in linear time in $|\Sigma||\mathcal{A}|$.

For the second property, we just need to check whether, for each $q \in Q$, the subautomaton induced by $[q]_x$ and transitions with priority $>x$ is strongly connected. ■

Case 1: $\mathcal{A}$ is already x-polished Assume that $\mathcal{A}$ is x-polished. In this case, it is almost an x-structured signature automaton. We just need to ensure that if $q \sim_x q'$, two transitions $q \xrightarrow{a:x} p$ and $q' \xrightarrow{a:x} p'$ go to a same state $p = p'$.

We remark that $\sim_x$ already satisfies most desired properties of monotonicity; only the uniformity for x-transitions is missing.

In order to obtain the strong congruence of x-transitions (Item 4), we redirect some x-transitions of $\mathcal{A}$. For each $\sim_x$-class $[q]_x$, pick an arbitrary state $f(q) \in [q]_x$. (Formally, $f : Q \to Q$ such that $f(q) = f(q')$ if $q \sim_x q'$). We let $\mathcal{A}'$ be the automaton obtained as follows:

- The states of $\mathcal{A}'$ are the same than those in $\mathcal{A}$.
- Transitions using priorities different from x are those in $\mathcal{A}$.
- If $q \xrightarrow{a:x} p$, we let $q \xrightarrow{a:x} f(p)$ in $\mathcal{A}'$.

It is immediate to check that $\mathcal{A}'$ is a $\sim_x$-nice transformation of $\mathcal{A}$ at level x. Moreover, $\mathcal{A}|_{\geq x+1} = \mathcal{A}'|_{\geq x+1}$. These remarks directly give:

LEMMA A.37. *$\mathcal{A}'$ is x-polished.*

LEMMA A.38. *There is a path $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ if and only if there is a path $q \overset{w':y}{\rightsquigarrow} p$ in $\mathcal{A}'$.*

PROOF. We suppose that there is $q \overset{w:y}{\rightsquigarrow} p$ in $\mathcal{A}$ (the converse proof is symmetric). If $y > x$, we have that $q \overset{w:y}{\rightsquigarrow} p$, as $\mathcal{A}|_{\geq x+1} = \mathcal{A}'|_{\geq x+1}$. If $y \leq x$, as $\mathcal{A}'$ is a nice transformation at level x, we have that $q \overset{w:y}{\rightsquigarrow} p'$ in $\mathcal{A}'$, with $p \sim_x p'$. As $\mathcal{A}'$ is x-polished, there is a path $p' \overset{w_2:>x}{\rightsquigarrow} p$. We conclude by taking $w' = ww_2$. ■

LEMMA A.39. *Automaton $\mathcal{A}'$ is equivalent to $\mathcal{A}$, and it is an x-structured signature automaton.*

PROOF. The fact that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ follows easily using that $\mathcal{A}'$ is a $\sim_x$-nice transformation of $\mathcal{A}$ at level x and applying Lemma A.6. Lemma A.38 implies that $\mathcal{A}'$ is in normal form. Verifying that $\mathcal{A}'$ is an x-structured signature automaton is just a routine check, using that $\mathcal{A}'$ is x-polished and Lemma A.38. ■

Case 2: Polishing a class Assume now that there is a class $[q]_x$ that is not x-polished in $\mathcal{A}$. We show that we can remove some states from this class, obtaining an strictly smaller equivalent automaton.

Local languages and local automata. We define the x-local alphabet at $[q]_x$ by

$$\Sigma_{[q]_x} = \{w \in \Sigma^+ \mid [q]_x \xrightarrow{w:\geq x} [q]_x \text{ and for any proper prefix } w' \text{ of } w, [q]_x \xrightarrow{w:\geq x} [p]_x \neq [q]_x\}.$$

We remark that, as $\sim_x$ is a congruence for $\geq x$-transitions (Lemma A.33), $\Sigma_{[q]_x}$ is well-defined and the notation $[q]_x \xrightarrow{w:\geq x}$ can be used. A word $w \in \Sigma^*$ belongs to $\Sigma^*_{[q]_x}$ if and only if it connects states in the class $[q]_x$. Elements in $\Sigma_{[q]_x}$ are those that do not pass twice through this class. Note that $\Sigma_{[q]_x}$ is a prefix code, and therefore it is uniquely decodable (even if, in general, it is infinite).

Seeing words in $\Sigma^\omega_{[q]_x}$ as words in Σ^ω, define the localisation of W to $[q]_x$ to be the objective

$$W_{[q]_x} = \{w \in \Sigma^\omega_{[q]_x} \mid w \in \mathcal{L}(\mathcal{A}_q)\}.$$

Observe that, as $\sim_x$ refines $\sim_{\mathcal{A}}$, this last definition does not depend on the choice of q and $W_{[q]_x}$ is prefix-independent. Moreover, $W_{[q]_x}$ is positional over finite, ε-free Eve-games: any $W_{[q]_x}$-game in which Eve could not play optimally using positional strategies would provide a counterexample for the positionality of $\mathcal{L}(\mathcal{A}_q)$, which is positional if W is (Lemma A.7).

The *local automaton of the class* $[q]_x$ is the automaton $\mathcal{A}_{[q]_x}$ defined as:

- The set of states is $[q]_x$.
- The initial state is arbitrary.
- For $w \in \Sigma_{[q]_x}$, $q_1 \xrightarrow{w:y} q_2$ if $q_1 \xrightarrow{w:y} q_2$ in $\mathcal{A}$ (we must have $y \geq x$).

Super words and super letters for local languages. We recall some terminology introduced in the warm-up. Assume that L is a prefix-independent language. We say that $u \in \Sigma^+$ is a *super word* for L if, for every $w \in \Sigma^\omega$, if w contains u infinitely often as a factor, then $w \in L$. If s is a letter, we say that it is a *super letter*.

For q a state and x an even priority, we let $B_{[q]_x} \subseteq \Sigma_{[q]_x}$ be the set of super letters for $W_{[q]_x}$, and we write $N_{[q]_x} = \Sigma_{[q]_x} \setminus B_{[q]_x}$. We refer to $N_{[q]_x}$ as the set of *neutral letters* of $\Sigma_{[q]_x}$ (for $W_{[q]_x}$).

LEMMA A.40 (Super words and uniformity). *A word $w \in \Sigma^\omega_{[q]_x}$ is a super word for $W_{[q]_x}$ if and only if w produces priority x uniformly in $[q]_x$, that is, for all $q' \in [q]_x$, $q' \xrightarrow{w:x} [q]_x$.*

PROOF. By normality of $\mathcal{A}$, if there is $q_1 \in [q]$ such that $q_1 \xrightarrow{w:>x} q_2$, there is a word $w' \in \Sigma^*$ labelling a returning path $q_2 \xrightarrow{w':x+1} q_1$. Therefore, $(ww')^\omega \notin \mathcal{L}(\mathcal{A}_q)$, so w is not a super word for $W_{[q]_x}$. The converse implication is clear, since each time word w is read, the minimal priority produced by the automaton is x. ∎

In particular, using previous lemma, we can detect the set of super letters $B_{[q]_x}$ in polynomial time.

Super words of positional languages. The use of the hypothesis of positionality of W for proving Lemma 5.22 resides in the next fundamental result.

LEMMA A.41 (Neutral letters do not form super words). *Let $w \in \Sigma^+_{[q]_x}$ be a super word for $W_{[q]_x}$. Then, w contains some super letter.*

PROOF. If w is already a letter in $\Sigma_{[q]_x}$, we are done. If not, let $w = w_1 w_2$ be any non-trivial decomposition into smaller words $w_1, w_2 \in \Sigma^+_{[q]_x}$. We show that either w_1 or w_2 are super words for $W_{[q]_x}$. This allows us to finish the proof, as we can recursively chop w into strictly smaller super words until obtaining a super letter.

Suppose by contradiction that neither w_1 or w_2 are super words. Then, by Lemma A.40, there are states q_1 and q_2 such that $q_1 \xrightarrow{w_1:>x} q_1'$ and $q_2 \xrightarrow{w_2:>x} q_2'$. By normality, we obtain returning paths $q_1' \xrightarrow{u_1:x+1} q_1$ and $q_2' \xrightarrow{u_2:x+1} q_2$. Therefore, $(w_1u_1)^\omega \notin W$ and $(w_2u_2)^\omega \notin W$. We consider the game $\mathcal{G}$ with winning condition $\mathcal{L}(\mathcal{A}_q)$ consisting in a vertex v with self loops u_1w_1 and u_2w_2 (see Figure 8 from the warm-up). Eve can win game $\mathcal{G}$, as alternating the two self loops she produces the word $(u_1w_1w_2u_2)^\omega$, which belongs to $\mathcal{L}(\mathcal{A}_q)$ since w_1w_2 is a super word. However, positional strategies in this game produce either $(w_1u_1)^\omega$ or $(w_2u_2)^\omega$, both losing. This contradicts the positionality of $\mathcal{L}(\mathcal{A}_q)$, and therefore, that of W (Lemma A.7). ■

Polishing a $\sim_x$-class of $\mathcal{A}$. We show how to polish the class $[q]_x$ of $\mathcal{A}$. This process has the property that, either $[q]_x$ is already polished, or the obtained automaton $\mathcal{A}'$ has strictly less states than $\mathcal{A}$, as desired.

Assume that the class $[q]_x$ is not x-polished. Consider the restriction of $\mathcal{A}_{[q]_x}$ to transitions labelled with $N_{[q]_x}$, which we denote $\mathcal{A}'_{[q]_x}$. Take $S_{[q]_x}$ to be a *final SCC* of $\mathcal{A}'_{[q]_x}$ (that is, one without edges leading to states not on itself).

Fix a state $q_0 \in [q]_x$. Consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by removing states in $[q]_x \setminus S_{[q]_x}$, and redirecting transition that go to $[q]_x \setminus S_{[q]_x}$ in $\mathcal{A}$ to transitions towards q_0. For these redirected transitions, we keep the same priority if it is $\leq x$, and set it to x otherwise. Formally:

- The set of states of $\mathcal{A}'$ is $Q' = Q \setminus \big([q]_x \setminus S_{[q]_x}\big)$.
- The initial state is q_{init}, or q_0 if $q_{\mathsf{init}} \in [q]_x \setminus S_{[q]_x}$.

For $q' \in Q'$:

- If $q' \xrightarrow{a:y} p$ in $\mathcal{A}$ and $p \notin [q]_x$, then $q \xrightarrow{a:y} p$ in $\mathcal{A}'$.
- If $q' \xrightarrow{a:y} p$ in $\mathcal{A}$, $p \in [q]_x \setminus S_{[q]_x}$, and $y \leq x$, then $q' \xrightarrow{a:y} q_0$ in $\mathcal{A}'$.
- If $q' \xrightarrow{a:y} p$ in $\mathcal{A}$, $p \in [q]_x \setminus S_{[q]_x}$, and $y > x$, then $q' \xrightarrow{a:x} q_0$ in $\mathcal{A}'$.

For transitions in the two latter cases, we say that $q' \xrightarrow{a:y} q_0$ has been *redirected* from $q' \xrightarrow{a:y} p$.

REMARK A.42. If $[q]_x = S_{[q]_x}$, then $\mathcal{A}' = \mathcal{A}$.

The following lemma will be use to show that we can compute $\mathcal{A}'$ in polynomial time, to prove the correctness of $\mathcal{A}'$, and to obtain that $[q]_x$ is x-polished in $\mathcal{A}'$.

LEMMA A.43. *Let $q_1, q_2 \in S_{[q]_x}$, and let $w \in N^*_{[q]_x}$ labelling a path $q_1 \overset{w:y}{\rightsquigarrow} q_2$ in $\mathcal{A}$. Then, $y > x$.*

PROOF. The fact that $y \geq x$ simply follows from the fact that $N_{[q]_x} \subseteq \Sigma_{[q]_x}$, which, by definition, contains words connecting the states in $[q]_x$ producing no priority $<x$.

Suppose by contradiction that $y = x$. Then, by the same argument as in the proof of Lemma 5.18 (see also Claim 4.15), there is $w' \in N^*_{[q]_x}$ producing priority x uniformly in $[q]_x$ and coming back to this class; that is, for every $q' \in [q]_x$, $q' \overset{w':x}{\rightsquigarrow} [q]_x$. Therefore, by Lemma A.40, w' is a super word. By Lemma A.41, w' must contain a super letter, a contradiction, as $w' \in N^*_{[q]_x}$ and $\Sigma_{[q]_x}$ is uniquely decodable. ■

LEMMA A.44. *Automaton $\mathcal{A}'$ can be computed in polynomial time from $\mathcal{A}$.*

PROOF. To obtain $S_{[q]_x}$, we first build a finite representation of the restriction of $\mathcal{A}_{[q]_x}$ to neutral letters (we recall that, in general, $\mathcal{A}_{[q]_x}$ might have an infinite number of transitions). One way of doing that is to build the following graph G: for each pair of states $q_1, q_2 \in [q]_x$ and each $y > x$, we put an edge $q_1 \xrightarrow{y} q_2$ if there is a path from q_1 to q_2 producing y as minimal priority and not passing trough another state in $[q]_x$. By Lemma A.43, $S_{[q]_x}$ is a subgraph of G. To obtain the states in $S_{[q]_x}$ we just need to perform a decomposition in SCCs of G and take a final SCC of it.[14] ■

We consider $\mathcal{A}'$ equipped with the preorders $\sqsubseteq_0, \dots, \sqsubseteq_x$ inherited from $\mathcal{A}$.

LEMMA A.45. *Automaton $\mathcal{A}'$ is a $\sim_x$-nice transformation of $\mathcal{A}$ at level x.*

PROOF. We first note that, by Lemma A.33, $\sim_x$ is $[0, x-1]$-faithful in $\mathcal{A}$, so it makes sense to speak of a $\sim_x$-nice transformation at level x.

Automaton $\mathcal{A}'|_{\geq x+1}$ coincides with the subautomaton of $\mathcal{A}|_{\geq x+1}$ induced by states in Q'. Indeed, let $q'_1, q'_2 \in Q'$ and $q'_1 \xrightarrow{a:>x} q'_2$ in $\mathcal{A}$. As these states are in Q', $q'_2 \notin [q]_x \setminus S_{[q]_x}$, so the transition has not been redirected, and it appears in $\mathcal{A}'$. Conversely, all transitions producing a priority $> x$ in $\mathcal{A}'$ appear in $\mathcal{A}$.

14 If W is not positional, the procedure described here does provide a set of states $S_{[q]_x}$, but it might lead to an incorrect automaton $\mathcal{A}'$. If our objective is to decide the positionality of W, at the end of the procedure we need to check the equality $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$; if it does not hold, we can conclude that W is not positional.

We show that $\sim_x$ is $[0, x-1]$-faithful in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\le x-1}^{x-1}} = \mathcal{A}/_{\sim_{\le x-1}^{x-1}}$. Let $p_1, p_2 \in Q'$ such that $p_1 \sim_x p_2$, and let $p_1 \xrightarrow{a:y_1'} q_1'$ and $p_2 \xrightarrow{a:y_2'} q_2'$ be two transitions in $\mathcal{A}'$. Transitions that have not been redirected satisfy the congruence requirements, as they satisfy them in $\mathcal{A}$. Assume that the first of these transitions have been redirected from $p_1 \xrightarrow{a:y_1} q_1$ in $\mathcal{A}$. We have that $q_1' = q_0 \sim_x q_1$, so, $q_1' \sim_x q_2'$ by the congruence property in $\mathcal{A}$. If $y_1' < x$, then $y_1 = y_1'$ and the y_1-uniformity of transitions in $\mathcal{A}$ yields $y_1' = y_2'$. Therefore, we also have that $y_1' \ge x$ if and only if $y_2' \ge x$ This gives both the $[0, x-1]$-faithfulness in $\mathcal{A}'$ and the equality of the quotient automata.

As $\sim_x$ refines $\sim_{\mathcal{A}}$, the latter relation is also a congruence in $\mathcal{A}'$ and $\mathcal{A}/_{\sim_{\mathcal{A}}} = \mathcal{A}'/_{\sim_{\mathcal{A}}}$. ■

LEMMA A.46 (Correctness of the polishing operation). *Automaton $\mathcal{A}'$ recognises $\mathcal{L}(\mathcal{A})$.*

PROOF. Let $w \in \Sigma^\omega$. If w is accepted or rejected with a priority $y < x$ or $y > x$ in $\mathcal{A}'$, by Lemma A.6, $w \in \mathcal{L}(\mathcal{A})$ if and only if $w \in \mathcal{L}(\mathcal{A}')$. Suppose then that w is accepted with priority x in $\mathcal{A}'$. Let ρ' be the run over w in $\mathcal{A}'$. If ρ' eventually does not take any redirected transition, then it is eventually a run in $\mathcal{A}$, and we can conclude by Lemma A.1. Suppose that ρ' takes redirected transitions infinitely often; moreover, eventually all such transitions produce priority x. We decompose ρ' as follows:

$$\rho' = q_{\mathsf{init}} \overset{w_0}{\rightsquigarrow} p_0' \xrightarrow{a_0:x} q_0 \overset{w_1:\ge x}{\rightsquigarrow} p_1' \xrightarrow{a_1:x} q_0 \overset{w_1:\ge x}{\rightsquigarrow} p_2' \xrightarrow{a_2:x} q_0 \rightsquigarrow \dots,$$

where no priority $< x$ appears after p_0', each transition $p_i' \xrightarrow{a_1:x} q_0$ is a redirected one, and no redirected transition appears in paths $q_0 \overset{w_i:\ge x}{\rightsquigarrow}$, in particular, these paths appear in $\mathcal{A}$.

CLAIM A.47. *For each $i \ge 1$, the word $w_i a_i$ belongs to $\Sigma^+_{[q]_x}$ and is a super word for $W_{[q]_x}$.*

Proof. The word $w_i a_i$ connects two states in $[q]_x$ in $\mathcal{A}'$ producing no priority $< x$. Since $\mathcal{A}'$ is a $\sim_x$-nice transformation at level x, word $w_i a_i$ also connects states in $[q]_x$ in $\mathcal{A}$, without producing priorities $< x$. Therefore, it belongs to $\Sigma^+_{[q]_x}$.

Consider the path $q_0 \overset{w_i}{\rightsquigarrow} p_i' \xrightarrow{a_i} q'$ in $\mathcal{A}$. As we suppose that transition $p_i' \xrightarrow{a_i} q_0$ has been redirected in $\mathcal{A}'$, $q' \notin S_{[q]_x}$. Then, since $S_{[q]_x}$ is a final SCC of the restriction of $\mathcal{A}_{[q]_x}$ to $N_{[q]_x}$-transitions, $w_i a_i$ contains some factor that is a letter in $\Sigma_{[q]_x} \setminus N_{[q]_x}$. Such a factor is a super letter, so $w_i a_i$ is a super word. ◆

Consider the run over w in $\mathcal{A}$, that we divide following the decomposition of ρ':

$$\rho = q_{\mathsf{init}} \overset{w_0}{\rightsquigarrow} p_0 \xrightarrow{a_0:\ge x} q_1 \overset{w_1:\ge x}{\rightsquigarrow} p_1 \xrightarrow{a_1:\ge x} q_2 \overset{w_2:\ge x}{\rightsquigarrow} p_2 \xrightarrow{a_2:\ge x} q_3 \rightsquigarrow \dots.$$

As $\mathcal{A}'$ is a $\sim_x$-nice transformation at level x, $q_i \sim_x q_0$ for all i, and ρ does not produce any priority $< x$ from p_0. By Lemma A.40, as $w_i a_i$ is a super word, the path $q_i \overset{w_i a_i}{\rightsquigarrow} q_{i+1}$ produces priority x. Therefore, w is accepted by $\mathcal{A}$. ■

LEMMA A.48 (Polishing polishes). *The class $[q]_x$ is x-polished in $\mathcal{A}'$.*

PROOF. Let $q_1, q_2 \in Q'$ be two states in the class $[q]_x$. Assume that, for a word $w \in \Sigma^*$, the path $q_1 \xrightarrow{w:x} p_1$ produces priority x. As $\sim_x$ is $[0, x-1]$-faithful, $q_2 \xrightarrow{w:\geq x} p_2$. Suppose by contradiction that this latter path produces exclusively priorities $> x$. Then, this path also exists in $\mathcal{A}$, and by normality (of $\mathcal{A}$), there is a returning path $p_2 \xrightarrow{w':x+1} q_2$. We obtain therefore a path $q_1 \xrightarrow{ww':x} [q]_x$. However, in $\mathcal{A}'$, $[q]_x = S_{[q]_x}$, so, by Lemma A.43, ww' contains a super letter, so $(ww')^\omega \in \mathcal{L}(\mathcal{A}_q)$, contradicting the fact that there is a cycle $q_2 \xrightarrow{ww':x+1} q_2$.

The second property of the definition of an x-polished class is satisfied in $\mathcal{A}'$, as we have redirected all x-transitions entering in $[q]_x$ to the state q_0.

We show the third item. Let $q_1, q_2 \in [q]_x$. Since $[q]_x = S_{[q]_x}$ in $\mathcal{A}'$, there is a path $q_1 \xrightarrow{w} q_2$ for some $w \in N^*_{[q]_x}$. By Lemma A.43, this path produces exclusively priorities $>x$. ■

This lemma allows us to conclude. We have obtained a deterministic automaton $\mathcal{A}'$ that is equivalent to $\mathcal{A}$. We claim that $|\mathcal{A}'| < |\mathcal{A}|$. Indeed, if this was not the case, we would have that $[q]_x = S_{[q]_x}$, so, by Remark A.42, $\mathcal{A} = \mathcal{A}'$. By the previous Lemma A.48 this implies that $[q]_x$ was already x-polished in $\mathcal{A}$, a contradiction.

Discussion: Why not just continue polishing? We have just showed a method to x-polish a given class of $\mathcal{A}$. The natural continuation would be to polish the rest of classes, until obtaining an x-polished automaton, and then apply the first case. The main difficulty is that the polishing operation we have presented might break the normality of $\mathcal{A}$. Normality of automata is key in all the process (see for example Lemma A.40), so we cannot guarantee to be able to continue polishing the classes of $\mathcal{A}'$. We would need to be able to either show that $\mathcal{A}'$ is in normal form (for example, by having an analogous to Lemma A.38), or to show that we can normalise $\mathcal{A}'$ while maintaining the properties of being an $(x-2)$-structured signature automaton. We have not succeeded in ensuring these properties, although we believe that it should be possible to do so.