



HAL
open science

Explorable automata

Emile Hazard, Denis Kuperberg

► **To cite this version:**

| Emile Hazard, Denis Kuperberg. Explorable automata. 2022. hal-03669659v1

HAL Id: hal-03669659

<https://hal.science/hal-03669659v1>

Preprint submitted on 16 May 2022 (v1), last revised 8 Nov 2022 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 Explorable automata

2 **Emile Hazard** ✉

3 CNRS, LIP, ENS Lyon, France

4 **Denis Kuperberg** ✉ 

5 CNRS, LIP, ENS Lyon, France

6 — Abstract —

7 We define the class of explorable automata on finite or infinite words. This is a generalization
8 of Good-For-Games (GFG) automata, where this time non-deterministic choices can be resolved
9 by building finitely many simultaneous runs instead of just one. We show that recognizing GFG
10 automata among explorable ones is in PTIME, thereby giving a strong link between the two notions.
11 We then show that recognizing explorable automata is EXPTIME-complete, in the case of finite words
12 or Büchi automata. Additionally, we define the notion of ω -explorable automata on infinite words,
13 where countably many runs can be used to resolve the non-deterministic choices. We show that all
14 reachability automata are ω -explorable, but this is not the case for safety ones. We finally show
15 EXPTIME-completeness for ω -explorability of automata on infinite words, covering the safety and
16 co-Büchi acceptance conditions.

17 **2012 ACM Subject Classification** F.4.3 Formal languages

18 **Keywords and phrases** Nondeterminism, automata, complexity

19 **Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

20 **1** Introduction

21 In several fields of theoretical science, the tension between deterministic and non-deterministic
22 models is a source of fundamental open questions, and has led to important lines of research.
23 The most famous of this kind is the P vs NP question in complexity theory. This paper aims
24 at further investigating the frontier between determinism and non-determinism in automata
25 theory. Although Non-deterministic and Deterministic Finite Automata (NFA and DFA) are
26 known to be equivalent, many subtle questions remain about the cost of determinism, and a
27 deep understanding of non-determinism will be needed to solve them.

28 One of the approaches investigating non-determinism in automata is the study of Good-
29 For-Games (GFG) automata, introduced in [13]. An automaton is GFG if when reading
30 input letters one by one, its non-determinism can be resolved on-the-fly without any need to
31 guess the future. This constitutes a model that is intermediary between non-determinism and
32 determinism, and can sometimes bring the best of both worlds. Like deterministic automata,
33 GFG automata on infinite words retain good properties such as their soundness with respect
34 to composition with games, making them appropriate for use in Church synthesis algorithms
35 [13]. On the other hand, like non-deterministic automata, they can be exponentially more
36 succinct than deterministic ones [16]. There is a very active line of research trying to
37 understand the various properties of GFG automata, see *e.g.* [21, 22, 4, 6, 17, 11, 23] for
38 latest developments. Notice that GFG automata are also called *history-deterministic*, a
39 terminology introduced originally in the theory of regular cost functions [9]. The name
40 “history-deterministic” corresponds to the above intuition of solving non-determinism on-the-
41 fly, while “good-for-games” refers to sound composition with games. These two notions may
42 actually differ in some quantitative frameworks, but coincide on boolean automata [5].

43 The goal of this paper is to pursue this line of research by introducing and studying the
44 class of explorable automata on finite and infinite words. The intuition behind explorability
45 is to limit the amount of non-determinism required by the automaton to accept its language,



© Emile Hazard and Denis Kuperberg;
licensed under Creative Commons License CC-BY 4.0
42nd Conference on Very Important Topics (CVIT 2016).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:26



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

46 in a more permissive way than GFG automata. If $k \in \mathbb{N}$, an automaton is k -explorable if
 47 when reading input letters, it suffices to keep track of k runs to build an accepting one, if it
 48 exists. An automaton is explorable if it is k -explorable for some $k \in \mathbb{N}$. This can be seen
 49 as a variation on the notion of GFG automaton, which corresponds to the case $k = 1$. The
 50 present work can be compared to [15], where a notion related to k -explorability (called *width*)
 51 is introduced and studied. In [15], the notion of simultaneous runs is different and more
 52 permissive, and does not give any meaningful notion of explorability, because n simultaneous
 53 runs always suffice for an automaton with n states. However some results of [15] also apply
 54 to k -explorability, notably EXPTIME-completeness of deciding k -explorability of an NFA if
 55 k is part of the input. Surprisingly however, the techniques used in [15] are quite different
 56 from the ones we need here. This shows that fixing a bound k for the number of runs leads
 57 to very different problems compared to asking for the existence of such a bound.

58 One of the motivations to introduce the notion of explorability is to tackle one of the
 59 important open questions about GFG automata: what is the complexity of deciding whether
 60 an automaton is GFG? Recognizing GFG automata is known to be in PTIME for Büchi [1]
 61 and co-Büchi [16] automata, but even for 3 parity ranks, the only known upper bound is
 62 EXPTIME via the naive algorithm from [13]. We show how explorable automata can simplify
 63 this question: if the input automaton is explorable, then the problem becomes PTIME.
 64 Therefore, the question of recognizing GFG automata can be shifted to: how hard is it to
 65 recognize explorable automata?

66 We then proceed to study the decidability and complexity of the explorability problem:
 67 deciding whether an input automaton on finite or infinite words is explorable. For this, we
 68 establish a connection with the population control problem studied in [2]. This problem
 69 asks, given an NFA with an arbitrary number of tokens in the initial state, whether a
 70 controller can choose input letters, thereby forcing every token to reach a designated state,
 71 even if tokens are controlled by an opponent. It is shown in [2] that the population control
 72 problem is EXPTIME-complete, and we adapt their proof to our setting to show that the
 73 explorability problem is EXPTIME-complete as well, already for NFAs. We also show that a
 74 direct reduction is possible, but at an exponential cost, yielding only a 2-EXPTIME algorithm
 75 for the NFA explorability problem. In the case of infinite words, we adapt the proof to the
 76 Büchi case, thereby showing that the Büchi explorability problem is in EXPTIME as well.
 77 We also remark that as in [2], the number of tokens needed to witness explorability can go
 78 as high as doubly exponential in the size of the automaton.

79 This EXPTIME-completeness result means that we unfortunately cannot directly use the
 80 intermediate notion of explorable automata to improve on the complexity of recognizing
 81 GFG automata, as could have been the hope. We still believe however that this explorability
 82 notion is of interest towards a better understanding of non-determinism in automata theory.

83 Notice that interestingly, from a model-checking perspective, our approach is dual to
 84 [2]: in the population control problem, an NFA is well-behaved when we can “control” it by
 85 forcing arbitrarily many runs to simultaneously reach a designated state, via an appropriate
 86 choice of input letters. On the contrary, in our approach, the input letters form an adversarial
 87 environment, and our NFA is well-behaved when its non-determinism is limited, in the sense
 88 that it is enough to spread finitely many runs to explore all possible behaviours.

89 On infinite words, we push further the notion of explorability, by remarking that for some
 90 automata, even following a countable number of runs is not enough. This leads to defining the
 91 class of ω -explorable automata, as those automata on infinite words where non-determinism
 92 can be resolved using countably many runs. We show that ω -explorable automata form a
 93 non-trivial class even for the safety acceptance condition (but not for reachability), and give

94 an EXPTIME algorithm recognizing ω -explorable automata, encompassing the safety and
 95 co-Büchi conditions. We also show EXPTIME-hardness of this problem, by adapting the
 96 EXPTIME-hardness proof of [2] to the setting of ω -explorability.

97 **Summary of the contributions.** We show that given an explorable parity automaton
 98 of fixed parity index, it is in PTIME to decide whether it is GFG. The algorithm used for
 99 Büchi in [1] is conjectured to work for any acceptance condition (this is the “ G_2 conjecture”),
 100 and it is in fact this algorithm that is shown here to work on any explorable parity automaton.

101 We show that given a NFA or Büchi automaton, it is decidable and EXPTIME-complete to
 102 check whether it is explorable. Our proof of EXPTIME-completeness for NFAs uses techniques
 103 developed in [2], where EXPTIME-completeness is shown for the NFA population control
 104 problem. We generalize this result to EXPTIME explorability checking for Büchi automata,
 105 requiring further adaptations. We also give a black box reduction using the result from
 106 [2]. This is enough to show decidability of the NFA explorability problem, but it yields a
 107 2-EXPTIME algorithm. As in [2], the EXPTIME algorithm yields a doubly exponential tight
 108 upper bound on the number of tokens needed to witness explorability.

109 On infinite words, we show that any reachability automaton is ω -explorable, but that this
 110 is not the case for safety automata. We show that both the safety and co-Büchi ω -explorability
 111 problems are EXPTIME-complete.

112 **Related Works.** Many works aim at quantifying the amount of non-determinism in
 113 automata. A survey by Colcombet [10] gives useful references on this question. Let us mention
 114 for instance the notion of ambiguity, which quantifies the number of simultaneous accepting
 115 runs. Similarly as in [15], we can note that ambiguity is orthogonal to k -explorability. Remark
 116 however that our finite/countable/uncountable explorability hierarchy is reminiscent of the
 117 finite/polynomial/exponential ambiguity hierarchy (see *e.g.* [24]).

118 In [14], several ways of quantifying the non-determinism in automata are studied from
 119 the point of view of complexity, including notions such as the number of advice bits needed.

120 Another approach is studied in [20], where a measure of the maximum non-deterministic
 121 branching along a run is defined and compared to other existing measures.

122 Following the GFG approach, a hierarchy of non-determinism and an analysis of this
 123 hierarchy via probabilistic models is given in [22].

124 We define explorability via games with tokens inspired by the approach in [1]. These
 125 games with tokens and their interplay with various quantitative acceptance conditions were
 126 recently investigated in [6].

127 2 Explorable automata

128 2.1 Preliminaries

129 If $i \leq j$ are integers, we will denote by $[i, j]$ the integer interval $\{i, i + 1, \dots, j\}$. If S is a set,
 130 its cardinal will be denoted $|S|$, and its powerset $\mathcal{P}(S)$.

131 We work with a fixed finite alphabet Σ . We will use the following default notation for the
 132 components of an automaton \mathcal{A} : $Q_{\mathcal{A}}$ for its states of states, $q_0^{\mathcal{A}}$ for its initial state, $F_{\mathcal{A}}$ for
 133 its accepting states, $\Delta_{\mathcal{A}}$ for its set of transitions. The subscript might be omitted when clear
 134 from context. We might also specify its alphabet by $\Sigma_{\mathcal{A}}$ instead of Σ for cases where different
 135 alphabets come into play. If $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $(p, a) \in Q \times \Sigma$,
 136 we will note $\Delta(p, a) = \{q \in Q, (p, a, q) \in \Delta\}$. If $X \subseteq Q$, we note $\Delta(X, a) = \bigcup_{p \in X} \Delta(p, a)$.

137 We will consider non-deterministic automata on finite words (NFAs). A run of such an
 138 automaton on a word $a_1 a_2 \dots a_n \in \Sigma^*$ is a sequence of states $q_0 q_1 \dots q_n \in Q^*$ (q_0 being the

139 initial state), such that for all $i \in [0, n - 1]$, we have $q_{i+1} \in \Delta(q_i, a_i)$. Such a run is accepting
 140 if $q_n \in F$, *i.e.* if the run belongs to Q^*F . As usual, the language of an automaton \mathcal{A} , denoted
 141 $L(\mathcal{A})$, is the set of words that admit an accepting run.

142 We will also deal with automata on infinite words, and we recall here some of the standard
 143 acceptance conditions for such automata. A run on an infinite word $w \in \Sigma^\omega$ is now an infinite
 144 sequence of states, *i.e.* an element of Q^ω , starting in q_0 and following as before transitions
 145 of the automaton according to the letters of w . Such a run of Q^ω is accepting in a safety
 146 (resp. reachability, Büchi, co-Büchi) automaton if it belongs to F^ω (resp. Q^*FQ^ω , $(Q^*F)^\omega$,
 147 Q^*F^ω). States from F will be called Büchi states in Büchi automata, and states from $Q \setminus F$
 148 will be called co-Büchi states in co-Büchi automata.

149 Finally, we will also mention the parity acceptance condition: it uses a ranking function
 150 rk from Q to an interval of integers $[i, j]$. A run is accepting if the minimal rank appearing
 151 infinitely often is even (following the convention of [2]).

152 2.2 Explorability

153 We start by introducing the *k-explorability game*, which is the central tool allowing us to
 154 define the class of explorable automata.

155 ► **Definition 1** (*k-explorability game*). Consider a non-deterministic automaton \mathcal{A} on finite
 156 or infinite words, and an integer k . The *k-explorability game* on \mathcal{A} is played on the arena
 157 Q^k . The two players are called *Determiniser* and *Spoiler*, and they play as follows.

- 158 ■ The initial position is the *k-tuple* $S_0 = (q_0, \dots, q_0)$.
- 159 ■ At step i from a position $S_{i-1} \in Q^k$, *Spoiler* chooses a letter $a_i \in \Sigma$, and *Determiniser*
 160 chooses $S_i \in Q^k$ such that for any token $l \in [0, k - 1]$, $S_{i-1}(l) \xrightarrow{a_i} S_i(l)$ is a transition of
 161 \mathcal{A} (where $S_i(l)$ stands for the l -th component in S_i).

162 The play is won by *Determiniser* if for any $\beta \leq \omega$ such that the word $(a_i)_{1 \leq i < \beta}$ is in $\mathcal{L}(\mathcal{A})$,
 163 there is a token $l < k$ being accepted by \mathcal{A} , meaning that the sequence $(S_i(l))_{i < \beta}$ is an
 164 accepting run¹. Otherwise the winner is *Spoiler*.

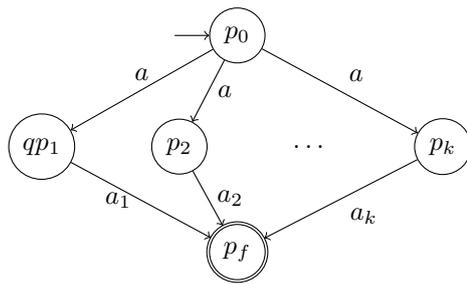
165 We will say that \mathcal{A} is *k-explorable* if *Determiniser* wins the *k-explorability game*.

166 We will say that \mathcal{A} is *explorable* if it is *k-explorable* for some $k \in \mathbb{N}$.

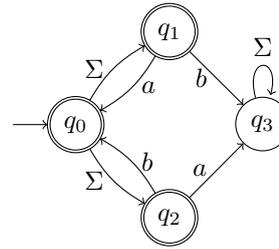
167 ► **Example 2.** The NFA \mathcal{A}_k on alphabet $\{a, a_1, \dots, a_k\}$ is *k-explorable* but not $(k - 1)$ -
 168 *explorable*. It can easily be adapted to a binary alphabet, by replacing in the automaton
 169 a_1, \dots, a_k by distinct words of the same length.

170 On the other hand, the NFA \mathcal{C} is a non-explorable NFA accepting all words on alphabet
 171 $\Sigma = \{a, b\}$. Indeed, *Spoiler* can win the *k-explorability game* for any k , by eliminating tokens
 172 one by one, choosing at each step the letter b if q_1 is occupied by at least one token, and the
 173 letter a otherwise.

¹ This condition $\beta \leq \omega$ is actually accounting separately for the two cases of finite and infinite words, corresponding respectively to $\beta < \omega$ and $\beta = \omega$.



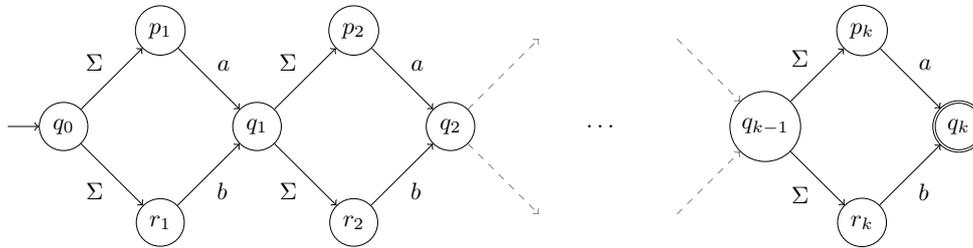
Explorable \mathcal{A}_k



Non-explorable \mathcal{C}

174

175 **► Example 3.** The following NFA \mathcal{B}_k with $3k + 1$ states on alphabet $\Sigma = \{a, b\}$ is explorable
 176 but requires 2^k tokens. Indeed, since when choosing the $2i^{\text{th}}$ letter Spoiler can always pick
 177 the state p_i or r_i containing the least amount of tokens to decide whether to play a or b , the
 178 best strategy for Determiniser is to split his tokens evenly at each q_i . This means he needs
 179 to start with 2^k tokens to end up with at least one token in q_k after a word of Σ^{2^k} .



180

181 Let us mention a few facts that follow from the definition of explorability:

182 **► Lemma 4.**

- 183 \blacksquare Any finite language is explorable.
- 184 \blacksquare If \mathcal{A} is k -explorable, then it is n -explorable for all $n \geq k$.
- 185 \blacksquare If \mathcal{A} is k -explorable and \mathcal{B} is n -explorable, then
 - 186 \blacksquare $\mathcal{A} \cup \mathcal{B}$ (with states $Q = \{q_0\} \cup Q_{\mathcal{A}} \cup Q_{\mathcal{B}}$) is $(k + n)$ -explorable,
 - 187 \blacksquare the union product $\mathcal{A} \times \mathcal{B}$ (with $F = (F_{\mathcal{A}} \times Q_{\mathcal{B}}) \cup (Q_{\mathcal{A}} \times F_{\mathcal{B}})$) is $\max(k, n)$ -explorable,
 - 188 \blacksquare the intersection product $\mathcal{A} \times \mathcal{B}$ (with $F = F_{\mathcal{A}} \times F_{\mathcal{B}}$) is (kn) -explorable.

189 **Proof.** If $L(\mathcal{A})$ is finite, it is enough to take $k = |L(\mathcal{A})|$ tokens to witness explorability: for
 190 each $u \in L(\mathcal{A})$, the token t_u assumes that the input word is u and follows an accepting run
 191 of \mathcal{A} over u as long as input letters are compatible with u . As soon as an input letter is not
 192 compatible with u , the token t_u is discarded and behaves arbitrarily for the rest of the play.

193 If \mathcal{A} is k -explorable and $n \geq k$, then Determiniser can win the n -explorability game by
 194 using the same strategy with the first k tokens and making arbitrary choices with the $n - k$
 195 remaining tokens.

196 If \mathcal{A} and \mathcal{B} are k - and n -explorable respectively, then Determiniser can use both strategies
 197 simultaneously with $k + n$ tokens in $\mathcal{A} \cup \mathcal{B}$, using k tokens in \mathcal{A} and n tokens in \mathcal{B} . If the
 198 input word is in \mathcal{A} (resp. \mathcal{B}), then the tokens playing in \mathcal{A} (resp. \mathcal{B}) will win the play.

199 In the union product $\mathcal{A} \times \mathcal{B}$, it is enough to take $\max(k, n)$ tokens: if $0 \leq i < \min(k, n)$,
 200 the token number i follows the strategy of the token i in \mathcal{A} on the first coordinate, and
 201 the strategy of the token i in \mathcal{B} in the second one. If $\min(k, n) \leq i < \max(k, n)$, say wlog
 202 $k \leq i < n$, the token i follows an arbitrary strategy on the \mathcal{A} -component and the strategy of
 203 token i on the \mathcal{B} -component.

204 However, Determiniser may need up to kn tokens to play in $\mathcal{A} \times \mathcal{B}$ when the accepting
 205 set is $F_{\mathcal{A}} \times F_{\mathcal{B}}$: the token (i, j) will use the strategy of the token i in the k -explorability
 206 game of \mathcal{A} together with the strategy of the token j in the n -explorability game of \mathcal{B} . This
 207 lower bound of kn cannot be improved: consider for instance $\mathcal{A}_k \times \mathcal{A}_n$, where $\mathcal{A}_k, \mathcal{A}_n$ are
 208 from Example 2. ◀

209 Notice that a similar notion was introduced in [15] under the name *width*. In [15], the
 210 emphasis is put on another version of the explorability game, where tokens can be duplicated,
 211 and $|Q|$ is an upper bound for the number of necessary tokens. In this work we will on the
 212 contrary focus on non-duplicable tokens, for which some results of [15] already apply. In
 213 particular the following holds:

214 ▶ **Theorem 5** ([15, Rem. 6.9]). *Given an NFA \mathcal{A} and an integer k , it is EXPTIME-complete*
 215 *to decide whether \mathcal{A} is k -explorable (even if we fix $k = |Q_{\mathcal{A}}|/2$).*

216 We aim here at answering a different question:

217 ▶ **Definition 6** (Explorability problem). *The explorability problem is the question, given a*
 218 *non-deterministic automaton \mathcal{A} , of deciding whether it is explorable.*

219 **Questions :** Is the explorability problem decidable ? If yes, what is its complexity ?
 220 We will first give some motivation for this problem in Section 2.3.

221 2.3 Link with GFG automata

222 An automaton \mathcal{A} is Good-for-Games (GFG) if it is 1-explorable, *i.e.* if there is a strategy
 223 $\sigma : \Sigma^* \rightarrow Q$ resolving the non-determinism based on the word read so far, with the guarantee
 224 that the run piloted by this strategy is accepting whenever the input word is in $L(\mathcal{A})$. See
 225 *e.g.* [3] for an introduction to GFG automata.

226 We will give here an additional and stronger link between explorable and GFG automata.
 227 In this part we will mainly be interested in automata on infinite words.

228 One of the main open problems related to GFG automata on infinite words is to decide,
 229 given a nondeterministic parity automaton, whether it is GFG. For now, the problem is only
 230 known to be in PTIME for co-Büchi [16] and Büchi [1] automata. Extending this result even
 231 to 3 parity ranks is still open, and only a naive EXPTIME upper bound [13] is known in this
 232 case. The following result shows that explorability is relevant in this context:

233 ▶ **Theorem 7.** *Given an explorable parity automaton \mathcal{A} of fixed parity index, it is in PTIME*
 234 *to decide whether it is GFG.*

235 This is one of the motivations to get a better understanding of explorable automata.
 236 Indeed, if we can obtain an efficient algorithm for recognizing them, or if we are in a context
 237 guaranteeing that we are only dealing with explorable automata, this result shows that we
 238 can obtain an efficient algorithm for recognizing GFG automata.

239 The rest of this section will be devoted to give a proof sketch of Theorem 7. See Appendix
 240 A.1 for formal details. The proof idea is inspired by [1].

241 Let \mathcal{A} be an explorable parity automaton, of fixed parity index $[i, j]$.

242 We briefly recall the definition of the game $G_k(\mathcal{A})$ defined in [1], for an arbitrary $k \in \mathbb{N}$.
 243 At each round, Adam plays a letter $a \in \Sigma$, then Eve moves her token according to an
 244 a -transition, and finally Adam moves his k tokens according to a -transitions. Eve wins the
 245 play if her token builds an accepting run, of if all of Adam's tokens build a rejecting run.

246 We will prove that the game $G_2(\mathcal{A})$ is won by Eve if and only if \mathcal{A} is GFG. Since $G_2(\mathcal{A})$
 247 can be solved in PTIME for fixed parity index [1], this is enough to conclude.

248 First, it is clear that if \mathcal{A} is GFG, then Eve wins $G_2(\mathcal{A})$ [1]: Eve can simply play her
 249 GFG strategy with her token, ignoring Adam's tokens.

250 For the converse, assume Eve wins $G_2(\mathcal{A})$, we want to prove that \mathcal{A} is GFG. We use the
 251 following lemma:

252 ► **Lemma 8** ([1, Thm. 14]). *Eve wins $G_2(\mathcal{A})$ if and only if Eve wins $G_k(\mathcal{A})$ for all $k \geq 2$.*

253 Since \mathcal{A} is explorable, there is $k \in \mathbb{N}$ such that \mathcal{A} is k -explorable. Let τ_k be a winning
 254 strategy for Determiniser in the k -explorability game of \mathcal{A} , and σ_k be a winning strategy for
 255 Eve in $G_k(\mathcal{A})$. We show that we can combine these two strategies to yield a GFG strategy σ
 256 for \mathcal{A} . This proof follows the same idea as in [1] where the explorability hypothesis is not
 257 available, but \mathcal{A} is assumed to be Büchi. The strategy σ will store k virtual tokens in its
 258 memory. When the automaton reads a new letter $a \in \Sigma$, these k tokens will be updated
 259 according to τ_k . Then the choice of σ will follow the strategy σ_k against these k tokens.
 260 Notice that the strategies τ_k and σ_k might use additional memory, but this is completely
 261 transparent in this proof scheme. If the input word is in $L(\mathcal{A})$, then by correctness of τ_k , one
 262 of the k virtual tokens will accept. Thus, by correctness of σ_k , the run chosen by σ will be
 263 accepting. Therefore, σ is a correct GFG strategy, witnessing that \mathcal{A} is GFG. This concludes
 264 the proof sketch of Theorem 7.

265 **3 Decidability and complexity of the explorability problem**

266 In this section, we prove that the explorability problem is decidable and EXPTIME-complete.

267 We start by showing in Section 3.1 decidability of the explorability problem for NFAs
 268 using the results of [2] as a black box. This yields an algorithm in 2-EXPTIME. We give
 269 in Section 3.2 a polynomial reduction in the other direction, thereby obtaining EXPTIME-
 270 hardness of the NFA explorability problem. To obtain a matching upper bound and show
 271 EXPTIME-completeness, we use again [2], but this time we must “open the black box” and
 272 dig into the technicalities of their EXPTIME algorithm while adapting them to our setting.
 273 We do so in Section 3.3, directly treating the more general case of Büchi automata.

274 **3.1 2-ExpTime algorithm via a black box reduction**

275 Let us start by recalling the population control problem (PCP) of [2].

276 ► **Definition 9** (k -population game). *Given an NFA \mathcal{B} with a distinguished target state $f \in Q_{\mathcal{B}}$,
 277 and an integer $k \in \mathbb{N}$, the k -population game is played similarly as the k -explorability game,
 278 only the winning condition differs: Spoiler wins if the game reaches a position where all
 279 tokens are in the state f .*

280 The PCP asks, given \mathcal{B} and $f \in Q_{\mathcal{B}}$, whether Spoiler wins the k -population game for all
 281 $k \in \mathbb{N}$. Notice that this convention is opposite to explorability, where positive instances are
 282 defined via a win of Determiniser. The PCP is shown in [2] to be EXPTIME-complete. We
 283 will present here a direct exponential reduction from the explorability problem to the PCP.

284 Let \mathcal{A} be a NFA. Our goal is to build an exponential NFA \mathcal{B} with a distinguished state f
 285 such that (\mathcal{B}, f) is a negative instance of the PCP if and only if \mathcal{A} is explorable.

286 We choose $Q_{\mathcal{B}} = (Q_{\mathcal{A}} \times \mathcal{P}(Q_{\mathcal{A}})) \uplus \{f, \perp\}$, where f, \perp are fresh sink states. The alphabet
 287 of \mathcal{B} will be $\Sigma_{\mathcal{B}} = \Sigma \uplus \{a_{\text{test}}\}$, where a_{test} is a fresh letter.

288 The initial state of \mathcal{B} is $q_0^{\mathcal{B}} = (q_0^{\mathcal{A}}, \{q_0^{\mathcal{A}}\})$. Notice that we do not need to specify accepting
 289 states in \mathcal{B} , as acceptance play no role in the PCP.

290 We finally define the transitions of \mathcal{B} in the following way:

- 291 ■ $(p, X) \xrightarrow{a} (q, \Delta_{\mathcal{A}}(X, a))$ if $a \in \Sigma$ and $q \in \Delta_{\mathcal{A}}(p, a)$,
- 292 ■ $(p, X) \xrightarrow{a_{\text{test}}} f$ if $p \notin F_{\mathcal{A}}$ and $X \cap F_{\mathcal{A}} \neq \emptyset$.
- 293 ■ $(p, X) \xrightarrow{a_{\text{test}}} \perp$ otherwise.

294 We aim at proving the following Lemma:

295 ► **Lemma 10.** *For any $k \in \mathbb{N}$, \mathcal{A} is k -explorable if and only if Determiniser wins the
 296 k -population game on (\mathcal{B}, f) .*

297 Notice that as long as letters of Σ are played, the second component of states of \mathcal{B} evolves
 298 deterministically and keeps track of the set of reachable states in \mathcal{A} . Moreover, the letter
 299 a_{test} also acts deterministically on $Q_{\mathcal{B}}$. Therefore, the only non-determinism to be resolved in
 300 \mathcal{B} is how the first component evolves, which amounts to building a run in \mathcal{A} . Thus, strategies
 301 driving tokens in \mathcal{A} and \mathcal{B} are isomorphic. It now suffices to observe that Spoiler wins the
 302 k -population game on (\mathcal{B}, f) if and only if he has a strategy allowing to eventually play a_{test}
 303 while all tokens are in a state of the form (q, X) with $q \notin F_{\mathcal{A}}$ and $X \cap F_{\mathcal{A}} \neq \emptyset$. This is
 304 equivalent to Spoiler winning the k -explorability game of \mathcal{A} , since $X \cap F_{\mathcal{A}} \neq \emptyset$ witnesses that
 305 the word played so far is in $L(\mathcal{A})$.

306 This concludes the proof that \mathcal{A} is explorable if and only if (\mathcal{B}, f) is a negative instance
 307 of the PCP. So given a NFA \mathcal{A} that we want to test for explorability, it suffices to build
 308 (\mathcal{B}, f) as above, and use the EXPTIME algorithm from [2] as a black box on (\mathcal{B}, f) . Since \mathcal{B}
 309 is of exponential size compared to \mathcal{A} , we obtain the following result:

310 ► **Theorem 11.** *The NFA explorability problem is decidable and in 2-EXPTIME.*

311 3.2 ExpTime-hardness of NFA explorability

312 We will perform here an encoding in the converse direction: starting from an instance (\mathcal{B}, f)
 313 of the PCP, we build polynomially a NFA \mathcal{A} such that \mathcal{A} is explorable if and only if (\mathcal{B}, f) is
 314 a negative instance of the PCP.

315 It is stated in [2] that without loss of generality, we can consider that f is a sink state in
 316 \mathcal{B} , and we will use this assumption here.

317 Let \mathcal{C} be the 4-state automaton of Example 2, that is non-explorable and accepts all
 318 words on alphabet $\Sigma_{\mathcal{C}} = \{a, b\}$. Recall that as an instance of the PCP, \mathcal{B} does not come with
 319 an acceptance condition. We will consider that its accepting set is $F_{\mathcal{B}} = Q_{\mathcal{B}} \setminus \{f\}$.

320 We will take for \mathcal{A} the product automaton $\mathcal{B} \times \mathcal{C}$ on alphabet $\Sigma_{\mathcal{A}} = \Sigma_{\mathcal{B}} \times \Sigma_{\mathcal{C}}$, with
 321 the union acceptance condition: \mathcal{A} accepts whenever one of its components accepts. The
 322 transitions of \mathcal{A} are defined as expected: $(p, p') \xrightarrow{a_1, a_2} (q, q')$ in \mathcal{A} whenever $p \xrightarrow{a_1} q$ in \mathcal{B} and
 323 $p' \xrightarrow{a_2} q'$ in \mathcal{C} .

324 Since $L(\mathcal{C}) = (\Sigma_{\mathcal{C}})^*$, we have $L(\mathcal{A}) = (\Sigma_{\mathcal{A}})^*$. The intuition for the role of \mathcal{C} in this
 325 construction is the following: it allows us to modify \mathcal{B} in order to accept all words, without
 326 interfering with its explorability status.

327 We claim that for any $k \in \mathbb{N}$, \mathcal{A} is k -explorable if and only if Determiniser wins the
 328 k -population game on (\mathcal{B}, f) .

329 Assume that \mathcal{A} is k -explorable, via a strategy σ . Then Determiniser can play in the
 330 k -population game on (\mathcal{B}, f) using σ as a guide. In order to simulate σ , one must feed to it
 331 letters from $\Sigma_{\mathcal{C}}$ in addition to letters from $\Sigma_{\mathcal{B}}$ chosen by Spoiler. This is done by applying

332 a winning strategy for Spoiler in the k -explorability game of \mathcal{C} . Assume for contradiction
 333 that at some point, this strategy σ reaches a position where all tokens are in a state of the
 334 form (f, q) with $q \in Q_{\mathcal{C}}$. Since f is a sink state, when the play continues it will eventually
 335 reach a point where all tokens are in (f, q_3) , where q_3 is the rejecting sink of \mathcal{C} . This is
 336 because we are playing letters from $\Sigma_{\mathcal{C}}$ according to a winning strategy for Spoiler in the
 337 k -explorability game of \mathcal{C} , and this strategy guarantees that all tokens eventually reach q_3 in
 338 \mathcal{C} . But this state (f, q_3) is rejecting in \mathcal{A} , and $L(\mathcal{A}) = (\Sigma_{\mathcal{A}})^*$, so this is a losing position for
 339 Determiniser in the k -explorability game of \mathcal{A} . Since we assumed σ is a winning strategy
 340 in this game, we reach a contradiction. This means that following this strategy σ together
 341 with an appropriate choice for letters from $\Sigma_{\mathcal{C}}$, we guarantee that at least one token never
 342 reaches the sink state f on its \mathcal{B} -component. This corresponds to Determiniser winning in
 343 the k -population game on (\mathcal{B}, f) .

344 Conversely, assume that Determiniser wins in the k -population game on (\mathcal{B}, f) , via a
 345 strategy σ . The same strategy can be used in the k -explorability game of \mathcal{A} , by making
 346 arbitrary choices on the \mathcal{C} component. As before, this corresponds to a winning strategy in
 347 the k -explorability game of \mathcal{A} , since there is always at least one token with \mathcal{B} -component in
 348 $F_{\mathcal{B}} = Q_{\mathcal{B}} \setminus \{f\}$. This achieves the hardness reduction, and allows us to conclude:

349 ► **Theorem 12.** *The NFA explorability problem is EXPTIME-hard.*

350 ► **Remark 13.** Using standard padding arguments, it is straightforward to extend Theorem 12
 351 to EXPTIME-hardness of explorability for automata on infinite words, using any of the
 352 acceptance conditions defined in Section 2.1.

353 Let us give some intuition on why we can obtain a polynomial reduction in one direction,
 354 but did not manage to do so in the other direction. Intuitively, the explorability problem is
 355 “more difficult” than the PCP for the following reason. In the PCP, Spoiler is allowed to play
 356 any letters, and the winning condition just depends on the current position. On the contrary,
 357 the winning condition of the k -explorability game mentions that the word chosen by Spoiler
 358 must belong to the language of the NFA. In order to verify this, we a priori need to append
 359 to the arena an exponential deterministic automaton for this language, and this is what is
 360 done in Section 3.1. This complicated winning condition is also the source of difficulty of
 361 recognizing GFG parity automata.

362 3.3 ExpTime algorithm for Büchi explorability

363 ► **Theorem 14.** *The explorability problem can be solved in EXPTIME for Büchi automata
 364 (and all simpler conditions).*

365 Due to space constraints we will only sketch the proof of Theorem 14 here. A more
 366 detailed account is given in Appendix A.2.

367 The algorithm is adapted from the EXPTIME algorithm for the PCP from [2]. We will
 368 recall here the main ideas of this algorithm, and describe how we adapt it to our setting.

369 Let \mathcal{A} be an NFA, together with a target state f . The idea in [2] is to abstract the
 370 population game with arbitrary many tokens by a game called the *capacity game*. This game
 371 allows Determiniser to describe only the support of his set of tokens, *i.e.* the set of states
 372 occupied by tokens. The sequence of states obtained in a play can be analyzed via a notion
 373 of *bounded capacity*, in order to detect whether it actually corresponds to a play with finitely
 374 many tokens. This notion can be approximated by the more relaxed *finite capacity*, which
 375 is a regular property that is equivalent to bounded capacity in a context where games are
 376 finite-memory determined. This property of finite capacity can be verified by a deterministic

377 parity automaton, yielding a parity game that can be won by Spoiler if and only if (\mathcal{A}, f) is
 378 a positive instance of the PCP. Since this parity game has size exponential in \mathcal{A} , this yields
 379 an EXPTIME algorithm for the PCP.

380 Here, we will perform the following tweaks to this construction. We now start with a
 381 Büchi automaton \mathcal{A} , and want to decide whether it is explorable.

382 First, we need to control that the infinite word played by Spoiler is in $L(\mathcal{A})$. This requires
 383 to build a deterministic parity automaton \mathcal{D} recognising $L(\mathcal{A})$, and incorporate it into the
 384 arena. The size of \mathcal{D} is exponential with respect to \mathcal{A} . We then follow [2] and build the
 385 capacity game augmented with \mathcal{D} . This time, a sequence of supports is winning if infinitely
 386 many of them contain an accepting state. We emphasize that we use here a particularity of
 387 the Büchi condition: observing the sequences of support sets of tokens is enough to decide
 388 whether one of the tokens follows an accepting run. The same particularity was used in [1],
 389 and was a crucial tool allowing to give a PTIME algorithm for Büchi GFGness. Since this
 390 modification still allows us to manipulate supports as simple sets, we can make use of the
 391 capacity game as before. We give in Appendix A.2, Remark 39 an example showing that a
 392 naïve adaptation of this construction to co-Büchi automata would not be correct.

393 Finally, we show that we can as in [2] obtain a parity game of exponential size character-
 394 izing explorability of \mathcal{A} , yielding the wanted EXPTIME algorithm.

395 We also remark that as in [2], this construction gives a doubly exponential upper bound
 396 on the number of tokens needed to witness explorability. Moreover, the proof from [2] that
 397 this is tight also stands here.

398 4 Explorability with countably many tokens

399 In this section we look at the same problem of explorability of an automaton, but we now
 400 allow for infinitely many tokens. More precisely, we will redefine the explorability game to
 401 allow an arbitrary cardinal for the number of tokens, then consider decidability problems
 402 regarding that game. This notion will mainly be interesting for automata on infinite words.

403 4.1 Definition and basic results

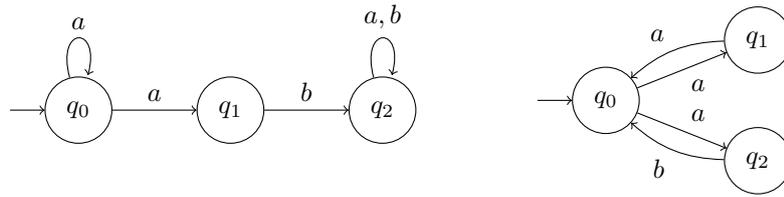
404 The following definition extends the notion of k -explorability to non-integer cardinals:

405 ► **Definition 15** (κ -explorability game). *Consider an automaton \mathcal{A} and a cardinal κ . The*
 406 *κ -explorability game on \mathcal{A} is played on the arena $(Q_{\mathcal{A}})^{\kappa}$, between Determiniser and Spoiler.*
 407 *They play as follows.*

- 408 ■ *The initial position is S_0 associating q_0 to all κ tokens.*
- 409 ■ *At step i , from position S_{i-1} , Spoiler chooses a letter $a_i \in \Sigma$, and Determiniser chooses*
 410 *S_i such that for any token α , $S_{i-1}(\alpha) \xrightarrow{a_i} S_i(\alpha)$ is a transition in \mathcal{A} .*

411 *The play is won by Determiniser if for any $\beta \leq \omega$ such that the word $(a_i)_{1 \leq i < \beta}$ is in $\mathcal{L}(\mathcal{A})$,*
 412 *there is a token $\alpha \in \kappa$ building an accepting run, meaning that the sequence $(S_i(\alpha))_{i < \beta}$ is an*
 413 *accepting run. Otherwise the winner is Spoiler.*

414 We will say in particular that \mathcal{A} is ω -explorable if Determiniser wins the game with
 415 ω tokens. We use here the notation ω for convenience, it should be understood as the
 416 countably infinite cardinal \aleph_0 . We will however explicitly use the fact that such an amount
 417 of tokens can be labelled by \mathbb{N} , in order to describe strategies for Spoiler or Determiniser
 418 in the ω -explorability game. The following lemma gives a first few results on generalised
 419 explorability.



■ **Figure 1** Two safety automata. Left: ω -explorable but not explorable. Right: not ω -explorable.

420 ► **Lemma 16.**

- 421 ■ *Determiniser wins the explorability game on \mathcal{A} with $|\mathcal{L}(\mathcal{A})|$ tokens.*
- 422 ■ *ω -explorability is not equivalent to explorability*
- 423 ■ *There are non ω -explorable safety automata.*

424 **Proof.** For the first item, a strategy for Determiniser is to associate a token to each word
 425 of $\mathcal{L}(\mathcal{A})$ and to have it follow an accepting run for that word. Let us add a few details
 426 on the cardinality of $L(\mathcal{A})$. First, a dichotomy result has been shown in [19] (even in the
 427 more general case of infinite trees): if $L(\mathcal{A})$ is not countable, then it has the cardinality
 428 of continuum, and this happens if and only if $L(\mathcal{A})$ contains a non-regular word. In this
 429 case, we can simply associate a token with every possible run. In the other case where
 430 $L(\mathcal{A})$ is countable, we have to associate an accepting run to each word, and this can be
 431 done without needing the Axiom of Countable Choice: a canonical run can be selected (*e.g.*
 432 lexicographically minimal).

433 We now want to prove that there are automata that are ω -explorable but not explorable.
 434 One such automaton is given in Figure 1 (left), where the rejecting sink state is omitted.
 435 Against any finite number of tokens, Spoiler has a strategy to eliminate them one by one,
 436 by playing a while Determiniser sends tokens to q_1 , and b the first time q_1 is empty after
 437 the play of Determiniser. On the other hand, with tokens indexed by ω , Determiniser can
 438 keep the token 0 in q_0 , and send token i to q_1 at step i . Those strategies are winning, which
 439 proves both non explorable and ω explorability of the automaton.

440 The last item is proven by the second example from Figure 1. A winning strategy for
 441 Spoiler against countable tokens consists in labelling the tokens with integers, then targeting
 442 each token one by one (first token 0, then 1, 2, *etc.*). Each token is removed using the correct
 443 two-letters sequence (a , then b if the token is in q_1 or a if it is in q_2). With this strategy,
 444 every token is removed at some point, even if there might always be tokens in the game. ◀

445 The first item of Lemma 16 implies that the ω -explorability game only gets interesting
 446 when we look at automata over infinite words: since any language of finite words over a finite
 447 alphabet is countable, Determiniser wins the corresponding ω -explorability game. We will
 448 therefore focus on infinite words in the following.

449 Let us emphasize the following slightly counter-intuitive fact: in the ω -explorability game,
 450 it is always possible for Determiniser to guarantee that infinitely many tokens occupy each
 451 currently reachable state. However, even in a safety automaton, this is not enough to win
 452 the game, as it does not prevent that each individual token might be eventually “killed” at
 453 some point. As the following Lemma shows, this phenomenon does not occur in reachability
 454 automata.

455 ► **Lemma 17.** *Any reachability automaton is ω -explorable.*

23:12 Explorable automata

456 **Proof.** For every $w \in \Sigma^*$ such that there is a finite run ρ leading to an accepting state,
457 Determiniser can use a single token following ρ . This token will accept all words of $w \cdot \Sigma^\omega$.
458 Since Σ^* is countable, we only need countably many such tokens to cover the whole language,
459 hence the result.

460 Let us give another equally simple view: a winning strategy for Determiniser in the
461 ω -explorability game is to keep infinitely many tokens in each currently reachable state, as
462 described above. Since acceptance in a reachability automaton is witnessed at a finite time,
463 this strategy is winning. ◀

4.2 ExpTime algorithm for co-Büchi automata

464 We already know, from the example of Figure 1, that the result from Lemma 17 does not
465 hold in the case of safety automata. However we have the following decidability result, which
466 talks about co-Büchi automata, and therefore still holds for safety automata as a subclass of
467 co-Büchi.
468

469 ▶ **Theorem 18.** *The ω -explorability of co-Büchi automata is decidable in EXPTIME.*

470 To prove this result, we will use the following *elimination game*. \mathcal{A} will from here on
471 correspond to a co-Büchi (complete) automaton. We start by building a deterministic
472 co-Büchi automaton \mathcal{D} for $L(\mathcal{A})$ (e.g. using the breakpoint construction [18]).

473 ▶ **Definition 19** (Elimination game). *The elimination game is played on the arena $\mathcal{P}(Q_{\mathcal{A}}) \times$
474 $Q_{\mathcal{A}} \times Q_{\mathcal{D}}$. The two players are named Protector and Eliminator, and the game proceeds as
475 follows, starting in the position $(\{q_0^{\mathcal{A}}\}, q_0^{\mathcal{A}}, q_0^{\mathcal{D}})$.*

- 476 ■ From position (B, q, p) Eliminator chooses a letter $a \in \Sigma$.
- 477 ■ If q is not a co-Büchi state, Protector picks a state $q' \in \Delta_{\mathcal{A}}(q, a)$.
- 478 ■ If q is a co-Büchi state, Protector picks any state $q' \in \Delta_{\mathcal{A}}(B, a)$. Such an event is called
479 elimination.
- 480 ■ The play moves to position $(\Delta_{\mathcal{A}}(B, a), q', \delta_{\mathcal{D}}(p, a))$.

481 Such a play can be written $(B_0, q_0, p_0) \xrightarrow{a_1} (B_1, q_1, p_1) \xrightarrow{a_2} (B_2, q_2, p_2) \dots$, and Eliminator
482 wins if infinitely many q_i and finitely many p_i are co-Büchi states.

483 Intuitively, what is happening in this game is that Protector is placing a token that
484 he wants to protect in a reachable state, and Eliminator aims at bringing that token to a
485 co-Büchi state while playing a word of $L(\mathcal{A})$. If Protector eventually manages to preserve his
486 token from elimination on an infinite suffix of the play, he wins.

487 ▶ **Lemma 20.** *The elimination game can be solved in polynomial time (in the size of the
488 game).*

489 **Proof.** To prove this result, we simply need to note that the winning condition is a parity
490 condition of fixed index. If we label the co-Büchi states q_i with rank 1, the co-Büchi states
491 p_i with rank 2, and the others with 3, then take the lowest rank in (B_i, q_i, p_i) (ignoring B_i),
492 Eliminator wins if and only if the inferior limit of ranks is even. As any parity game with 3
493 ranks can be solved in polynomial time [7], this is enough to get the result. ◀

494 We want to prove the equivalence between this game and the ω -explorability game to
495 obtain Theorem 18.

496 ▶ **Lemma 21.** *\mathcal{A} is ω -explorable if and only if Protector wins the elimination game on \mathcal{A} .*

497 **Proof.** First let us suppose that Eliminator wins the elimination game on \mathcal{A} . To build a
 498 strategy for Spoiler in the ω -explorability game of \mathcal{A} , we first take a function $f : \mathbb{N} \rightarrow \mathbb{N}$
 499 such that for any $n \in \mathbb{N}$, $|f^{-1}(n)|$ is infinite (for instance f is described by the sequence
 500 $0, 0, 1, 0, 1, 2, 0, 1, 2, 3, \dots$). The strategy for Spoiler will focus on sending token $f(0)$, then
 501 $f(1)$, then $f(2)$, *etc.* to a co-Büchi state.

502 Let σ be a memoryless winning strategy for Eliminator in the elimination game (recall
 503 that parity games do not require memory [12]). Spoiler will follow this strategy σ in the
 504 ω -explorability game, by keeping an imaginary play of the elimination game in his memory:
 505 $M = \mathcal{P}(Q_{\mathcal{A}}) \times Q_{\mathcal{A}} \times Q_{\mathcal{D}} \times \mathbb{N}$.

- 506 ■ At first the memory holds the initial state $(\{q_0^{\mathcal{A}}\}, q_0^{\mathcal{A}}, q_0^{\mathcal{D}}, 0)$, and the current target is
 507 given by the last component: it is the token $f(0)$.
- 508 ■ From (B, q, p, n) Spoiler plays in both games the letter a given by σ .
- 509 ■ Once Determiniser has played, Spoiler moves the memory to $(\Delta_{\mathcal{A}}(B, a), q', \delta_{\mathcal{D}}(p, a), n)$
 510 where q' is the new position of the token $f(n)$, except if q was a co-Büchi state, in which
 511 case we move to $(\Delta_{\mathcal{A}}(B, a), q', \delta_{\mathcal{D}}(p, a), n + 1)$ where q' is the new position of the token
 512 $f(n + 1)$. We then go back to the previous step.

513 This strategy builds a play of the elimination game in the memory, that is consistent with σ .
 514 We know that σ is winning, which implies that the word played is in $\mathcal{L}(\mathcal{A})$, and that every
 515 $n \in \mathbb{N}$ is visited (each elimination increments n , and there are infinitely many of those). An
 516 elimination happening while the target is the token $f(n)$ corresponds, on the exploration
 517 game, to that token visiting a co-Büchi state. Ultimately this means that Determiniser did
 518 not provide any accepting run, while Spoiler did play a word from $\mathcal{L}(\mathcal{A})$, and therefore won.

519 Let us now consider the situation where Protector wins the elimination game, using some
 520 strategy τ . We want to build a winning strategy for Determiniser in the ω -explorability
 521 game. Similarly, this strategy will keep track of a play in the elimination game in its memory.
 522 Determiniser will maintain ω tokens in any reachable state, while focusing on a particular
 523 token which follows the path of the current target in the elimination game. When that token
 524 visits a co-Büchi state, we switch to the new token specified by τ .

525 Since τ is winning in the elimination game, either the word played by Spoiler is not in
 526 $\mathcal{L}(\mathcal{A})$, which ensures a win for Determiniser, or there are no eliminations after some point,
 527 meaning that the target token at that point never visits another co-Büchi state, which also
 528 implies that Determiniser wins. ◀

529 With Lemmas 20 and 21 we get a proof of Theorem 18, since the elimination game
 530 associated to \mathcal{A} is of exponential size and can be built using exponential time.

531 4.3 ExpTime-hardness of the ω -explorability problem

532 ▶ **Theorem 22.** *The ω -explorability problem for (any automaton model embedding) safety*
 533 *automata is EXPTIME-hard.*

534 We give a quick summary of the proof in this section. The full proof can be found
 535 in Appendix A.3. The main idea will be to reduce the acceptance problem of a PSPACE
 536 alternating Turing machine (ATM) to the ω -explorability problem of some automaton that
 537 we build from the machine. This reduction is an adaptation of the one from [2] showing
 538 EXPTIME-hardness of the NFA population control problem (defined in Section 3.1).

539 The computation of an ATM can be seen as a game between two players who respectively
 540 aim for acceptance and rejection of the input. These players influence the output by choosing
 541 the transitions when facing a non-deterministic choice, that can belong to either one of them.

542 Let us first describe the automaton built in [2]. In that reduction, the choices made by
 543 the ATM players are translated into choices for Determiniser and Spoiler. The automaton
 544 has two main blocks: one dedicated to keeping track of the machine's configuration, which
 545 we call Config, and another focusing on the simulation of the ATM choices, which we call
 546 Choices. In Config, there is no non-determinism: the tokens move following the transitions of
 547 the machine given as input to the automaton. In Choices, Determiniser can pick a transition
 548 by sending his token to the corresponding state, while Spoiler uses letters to pick his.

549 The automaton constructed this way will basically read a sequence of runs of the ATM.
 550 At each run, some tokens must be sent into both blocks. Reaching an accepting state of a
 551 run lets Spoiler send some tokens from Choices to his target state, specifically those whose
 552 choices for the transitions of the ATM were followed. He can then restart with the remaining
 553 tokens until all are in the target. This process will ensure a win for Spoiler if he has a winning
 554 strategy in the ATM game. If he does not, then Determiniser can use a strategy ensuring
 555 rejection in the ATM game to avoid the configurations where he loses tokens, provided he
 556 starts with enough tokens.

557 This equivalence between acceptance of the ATM and the automaton being a positive
 558 instance of the PCP provides the EXPTIME-hardness of their problem.

559 In our setup, getting rid of tokens one by one is not enough: Spoiler needs to be able to
 560 target a specific token and send it to the target state (which is now the rejecting state \perp)
 561 in one run. If he can do that, repeating the process for every token, without omitting any,
 562 ensures his win. If he cannot, then Determiniser has a strategy to pick a specific token and
 563 preserving it from \perp , and therefore wins.

564 This is why we adapt our reduction to allow Spoiler to target a specific token, no matter
 565 where it chooses to go. To do so, we change the transitions so that winning a run lets Spoiler
 566 additionally send every token from Config into \perp . With that and the fact that he can already
 567 target a token in Choices, we get a winning strategy for Spoiler when the ATM is accepting.

568 If the ATM is rejecting, Spoiler is still able to send some tokens to \perp , but he no longer
 569 has that targeting ability, which is how Determiniser is able to build a strategy preserving a
 570 specific token to win. To ensure the sustainability of this method, Determiniser needs to
 571 keep ω additional tokens following his designated token, so that he always has ω tokens to
 572 spread into the gadgets every time a new run starts.

573 Overall, we are able to compute in polynomial time from the ATM a safety automaton
 574 that is ω -explorable if and only if the ATM rejects its input. Since acceptance of a polynomial
 575 space ATM is known to be EXPTIME-hard, we obtain Theorem 22.

576 Conclusion

577 We introduced and studied the notions of explorability and ω -explorability, for automata on
 578 finite and infinite words. We showed that these problems are EXPTIME-complete for Büchi
 579 condition in the first case and co-Büchi condition in the second case.

580 It is plausible that these results could be generalised to higher parity conditions, for
 581 instance by replacing the notion of support set by Safra trees, but this is outside of the scope
 582 of this paper and we leave this investigation for further research.

583 Although we showed that the original motivation of using explorability to improve the
 584 current knowledge on the complexity of the GFGness problem for all parity automata cannot
 585 be directly achieved, since deciding explorability is at least as hard as GFGness, we believe
 586 that explorability is a natural property in the study of degrees of nondeterminism, and that
 587 this notion could be used in other contexts as a middle ground between deterministic and
 588 non-deterministic automata.

589 — **References** —

- 590 1 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable.
591 In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical*
592 *Computer Science, FSTTCS 2018*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik,
593 2018.
- 594 2 Nathalie Bertrand, Miheer Dewaskar, Blaise Genest, Hugo Gimbert, and Adwait Amit Godbole.
595 Controlling a population. *Log. Methods Comput. Sci.*, 15(3), 2019.
- 596 3 Udi Boker, Denis Kuperberg, Orna Kupferman, and Michal Skrzypczak. Nondeterminism in
597 the presence of a diverse or unknown future. In *Automata, Languages, and Programming -*
598 *40th International Colloquium, ICALP 2013*, Lecture Notes in Computer Science. Springer,
599 2013.
- 600 4 Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michal Skrzypczak. On the succinct-
601 ness of alternating parity good-for-games automata. In *40th IARCS Annual Conference on*
602 *Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020*, LIPIcs.
603 Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 604 5 Udi Boker and Karoliina Lehtinen. History determinism vs. good for gameness in quantitative
605 automata. In *41st IARCS Annual Conference on Foundations of Software Technology and*
606 *Theoretical Computer Science, FSTTCS 2021*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für
607 Informatik, 2021.
- 608 6 Udi Boker and Karoliina Lehtinen. Token games and history-deterministic quantitative
609 automata. In *Foundations of Software Science and Computation Structures - 25th International*
610 *Conference, FOSSACS 2022*, Lecture Notes in Computer Science. Springer, 2022.
- 611 7 Cristian S. Calude, Sanjay Jain, Bakhadyr Khossainov, Wei Li, and Frank Stephan. Deciding
612 parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT*
613 *Symposium on Theory of Computing, STOC 2017*, pages 252–263. ACM, 2017.
- 614 8 Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games.
615 In Helmut Seidl, editor, *Foundations of Software Science and Computational Structures, 10th*
616 *International Conference, FOSSACS 2007*, Lecture Notes in Computer Science. Springer, 2007.
- 617 9 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In
618 *Automata, languages and programming. Part II*, volume 5556 of *Lecture Notes in Comput.*
619 *Sci.*, pages 139–150, Berlin, 2009. Springer.
- 620 10 Thomas Colcombet. Forms of determinism for automata (invited talk). In *29th International*
621 *Symposium on Theoretical Aspects of Computer Science, STACS 2012*, LIPIcs. Schloss Dagstuhl
622 - Leibniz-Zentrum für Informatik, 2012.
- 623 11 Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata:
624 New tools for infinite duration games. In *Foundations of Software Science and Computation*
625 *Structures - 22nd International Conference, FOSSACS 2019*, Lecture Notes in Computer
626 Science. Springer, 2019.
- 627 12 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy
628 (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San*
629 *Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991.
- 630 13 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer*
631 *Science Logic, 20th International Workshop, CSL 2006*, 2006.
- 632 14 Juraj Hromkovic, Juhani Karhumäki, Hartmut Klauck, Georg Schnitger, and Sebastian Seibert.
633 Measures of nondeterminism in finite automata. *Electronic Colloquium on Computational*
634 *Complexity (ECCC)*, 7, 01 2000.
- 635 15 Denis Kuperberg and Anirban Majumdar. Computing the width of non-deterministic automata.
636 *Log. Methods Comput. Sci. (LMCS)*, 15(4), 2019.
- 637 16 Denis Kuperberg and Michal Skrzypczak. On determinisation of good-for-games automata.
638 In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015*,
639 Lecture Notes in Computer Science. Springer, 2015.

- 640 17 Karoliina Lehtinen and Martin Zimmermann. Good-for-games ω -pushdown automata. In *LICS*
641 *2020: 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 689–702.
642 ACM, 2020.
- 643 18 Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω -words. *Theoret. Comput.*
644 *Sci.*, 32(3):321–330, 1984.
- 645 19 Damian Niwinski. On the cardinality of sets of infinite trees recognizable by finite automata.
646 In *Mathematical Foundations of Computer Science 1991, 16th International Symposium,*
647 *MFCS'91*, Lecture Notes in Computer Science. Springer, 1991.
- 648 20 Alexandros Palioudakis, Kai Salomaa, and Selim G. Akl. Worst case branching and other
649 measures of nondeterminism. *Int. J. Found. Comput. Sci.*, 28(3):195–210, 2017.
- 650 21 Bader Abu Radi and Orna Kupferman. Minimization and canonization of GFG transition-based
651 automata. *CoRR*, abs/2106.06745, 2021.
- 652 22 Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz. A hierarchy of nondeterminism.
653 In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS*
654 *2021*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- 655 23 Sven Schewe. Minimising good-for-games automata is np-complete. In *40th IARCS Annual*
656 *Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*
657 *2020*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
- 658 24 Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor.*
659 *Comput. Sci.*, 88(2):325–349, 1991.

A Appendix

A.1 Link with GFG automata

We describe here in more detail how, assuming that \mathcal{A} is explorable and Eve wins $G_k(\mathcal{A})$ for some $k \in \mathbb{N}$, we obtain a GFG strategy for \mathcal{A} . Let us note $Q = Q_{\mathcal{A}}$ the set of states of \mathcal{A} .

In the proof sketch of Section 2.3, we defined τ_k to be a winning strategy for Determiniser in the k -explorability game, and σ_k a winning strategy for Eve in $G_k(\mathcal{A})$.

Let us explicit in detail the shape of these strategies. The strategy τ_k has access to the history of the play in the k -explorability game, and must decide on a move for Determiniser. Notice that it is always enough to know the history of the opponent's moves (here the letters of Σ played so far), since this allows to compute the answer of Determiniser at each step, and therefore build a unique play. Thus we can take for τ_k a function $\Sigma^* \rightarrow Q^k$. If the word played so far is $u \in \Sigma^*$, the tuple of states reached by the k tokens moved according to τ_k is $\tau_k(u) \in Q^k$, with in particular $\tau_k(\varepsilon) = (q_0^A, \dots, q_0^A)$.

If $w = a_1 a_2 \dots \in \Sigma^\omega$, and $i \in \mathbb{N}$, let us note $(q_{w,1}^i, \dots, q_{w,k}^i) = \tau_k(a_1 \dots a_i)$. That is $q_{w,j}^i$ is the state reached by the j^{th} token after i steps in the run induced by τ_k and u . If $j \in [1, k]$, let us note $\rho_{u,j}$ the infinite run $q_{w,j}^0 q_{w,j}^1 q_{w,j}^2 \dots$, followed by the j^{th} token in this play. By definition of τ_k , we have the guarantee that for all $w \in L(\mathcal{A})$, there exists $j \in [1, k]$ such that $\rho_{w,j}$ is accepting.

If $u = a_1 \dots a_n \in \Sigma^*$ is a finite word, we define $\tau'_k(u) = (\tau_k(\varepsilon), \tau_k(a_1), \tau_k(a_1 a_2) \dots, \tau_k(u))$ the list of partial runs induced by τ_k on u .

Let us now turn to the strategy σ_k of Eve in $G_k(\mathcal{A})$. The type of this strategy is $\sigma_k : \Sigma^* \times (Q^k)^* \rightarrow Q$. Indeed, this time, the history of Adam's moves must contain his choice of letters together with his choices of positions for his k tokens. So $\sigma_k(u, \gamma)$ gives the state reached by Eve's token after an history (u, γ) for the moves of Adam. Notice that at each step, Eve must move before Adam in this game $G_k(\mathcal{A})$, so γ does not contain the choice of Adam on the last letter of u . This means that except for $u = \varepsilon$, we can always assume $|u| = |\gamma| + 1$ in a history (u, γ) .

We have the guarantee that if Adam plays an infinite word w together with runs ρ_1, \dots, ρ_k on w , at least one of which is accepting, then the run yielded by σ_k against $(w, (\rho_1, \dots, \rho_k))$ is accepting.

We finally define the GFG strategy σ for \mathcal{A} , of type $\Sigma^* \rightarrow Q$, by induction: $\sigma(\varepsilon) = q_0^A$, and $\sigma(ua) = \sigma_k(ua, \tau'_k(u))$.

This amounts to playing the strategy σ_k in $G_k(\mathcal{A})$, against Adam playing a word w and moving his k tokens according to the strategy τ_k against w . If the infinite word $w = a_1 a_2 \dots$ chosen by Adam is in $L(\mathcal{A})$, then by correctness of τ_k one of the k runs $\rho_{w,1}, \dots, \rho_{w,k}$ yielded by τ_k is accepting. Hence, by correctness of σ_k , the run $\sigma(\varepsilon)\sigma(a_1)\sigma(a_1 a_2)$ yielded by σ (based on σ_k) is accepting. This concludes the proof that σ is a correct GFG strategy for \mathcal{A} , witnessing that \mathcal{A} is GFG.

A.2 ExpTime algorithm for Büchi explorability

We will prove here Theorem 14 from Section 3.3.

In this part, $\mathcal{A} = (\Sigma, Q, q_0^A, \Delta_{\mathcal{A}}, F_{\mathcal{A}})$ is a non-deterministic Büchi automaton. We start by computing in exponential time an equivalent deterministic parity automaton $\mathcal{D} = (\Sigma, Q_{\mathcal{D}}, q_0^{\mathcal{D}}, \delta_{\mathcal{D}}, F_{\mathcal{D}})$, via any standard method.

The algorithm described in this section is adapted from [2]. Many results from this previous work still hold in our framework. We will however need to adapt some constructions

23:18 Explorable automata

705 and give new arguments, both to fit our explorability framework, and to generalize from
706 NFA to Büchi automata.

707 ► **Definition 23** (Transfer graph). *A transfer graph G is a subset of $Q \times Q$. We say that it*
708 *is compatible with a letter a if every edge in G corresponds to a transition in \mathcal{A} labelled by*
709 *a , i.e. for any $(q, r) \in G$, we have $(q, a, r) \in \Delta_{\mathcal{A}}$. In other words, G is a subgraph of the*
710 *transition graph of the letter a .*

711 *Given a transfer graph G and a set of states $X \subseteq Q$, we note $G(X) = \{q \in Q \mid \exists r \in$
712 $X, (q, r) \in G\}$. We call respectively $\text{Dom}(G)$ and $\text{Im}(G)$ the projections of G on its first and
713 second coordinate, i.e. $\text{Dom}(G) = \{q \in Q \mid \exists r \in Q, (q, r) \in G\}$ and $\text{Im}(G) = G(Q)$.*

714 *The composition of transfer graphs is defined the natural way: $G \cdot H = \{(x, z) \mid \exists y, (x, y) \in$
715 $G \wedge (y, z) \in H\}$.*

716 ► **Definition 24** (Support game). *The support game is played in the arena $\mathcal{P}(Q) \times Q_{\mathcal{D}}$, called*
717 *support arena. It is played as follows by Determiniser and Spoiler.*

- 718 ■ *The starting support is $S_0 = (\{q_0^A\}, q_0^{\mathcal{D}})$.*
- 719 ■ *At any given step with support (B, q) , Spoiler chooses a letter $a \in \Sigma$, then Determiniser*
720 *chooses a transfer graph G compatible with a , and with $\text{Dom}(G) = B$. The play then*
721 *moves to $(\text{Im}(G), \delta_{\mathcal{D}}(q, a))$.*

722 *A play can be represented by a sequence $(B_0, q_0) \xrightarrow{a_1, G_1} (B_1, q_1) \xrightarrow{a_2, G_2} (B_2, q_2) \dots$*

723 *We say that Spoiler wins the play if the run $q_0 q_1 q_2 \dots$ of \mathcal{D} is parity accepting, while only*
724 *finitely many B_i contain Büchi states (from $F_{\mathcal{A}}$).*

725 Note that a winning strategy for Determiniser in the support game cannot in general
726 be interpreted as a witness of explorability. This is illustrated by the automaton \mathcal{C} from
727 Example 2. For any $k \in \mathbb{N}$, the k -explorability game is won by Spoiler on that automaton,
728 while Determiniser wins the support game. Intuitively, the support game does not account
729 for the limits of resources for Determiniser.

730 On the other hand, a winning strategy for Spoiler in this support game does translate
731 into a non-explorability witness, i.e. a strategy for Spoiler in the k -explorability game for
732 any k . The support game is therefore “too easy” for Determiniser, and this is what we try to
733 correct in the following.

734 ► **Definition 25** (Projection of a play). *Given a play $S_0 \xrightarrow{a_1} S_1 \xrightarrow{a_2} S_2 \dots$ in the k -explorability*
735 *game, the projection of that play in the support arena is the play $(B_0, q_0) \xrightarrow{a_1, G_1} (B_1, q_1) \xrightarrow{a_2, G_2}$*
736 *$(B_2, q_2) \dots$, where:*

- 737 ■ *B_i is the support of S_i (states occupied in S_i),*
- 738 ■ *$q_0 = q_0^{\mathcal{D}}$ and $q_{i+1} = \delta_{\mathcal{D}}(a_{i+1}, q_i)$ for all i ,*
- 739 ■ *$G_{i+1} = \{(S_i(j), S_{i+1}(j)) \mid j \in [0, k-1]\}$.*

740 *This corresponds to forgetting the multiplicity of tokens and only keeping track of the transitions*
741 *that are used.*

742 ► **Definition 26** (Realisable play). *A play in the support arena is realisable if it is the projection*
743 *of a play in the k -explorability game for some $k \in \mathbb{N}$.*

744 We would like to restrict plays in the support arena to realisable ones only. To do so, we
745 define the notion of capacity as follows.

746 ► **Definition 27** (Accumulator and capacity [2]). *In a play $(B_0, q_0) \xrightarrow{a_1, G_1} (B_1, q_1) \xrightarrow{a_2, G_2}$*
 747 *$(B_2, q_2) \dots$, an accumulator is a sequence $(T_j)_{j \in \mathbb{N}}$ such that for any j , $T_j \subseteq B_j$ and $T_{j+1} \supseteq$*
 748 *$G_{i+1}(T_j)$. An edge $(q, r) \in G_{j+1}$ is an entry for $(T_j)_{j \in \mathbb{N}}$ at index i if $q \notin T_j$ and $r \in T_{j+1}$.*
 749 *A play has finite capacity if every accumulator has finitely many entries, and bounded*
 750 *capacity if the number of entries of its accumulators is bounded.*

751 This definition gives us tools to talk about realisable plays in a more practical way, as
 752 shown by the following Lemma. Note that although the explorability game is replaced by
 753 the population control game in [2], the same proof still applies here.

754 ► **Lemma 28** ([2, Lem 3.5]). *A play is realisable if and only if it has bounded capacity.*

755 Moreover, the proof of Lemma 28 can also be used to get the following result, which we
 756 will use later. Note that we talk about the explorability game in this Lemma, but this only
 757 concerns its arena regardless of the winning condition. The proof holds because the arena
 758 from [2] is identical.

759 ► **Lemma 29** ([2, Lem 3.5]). *If Determiniser has a strategy τ in the support arena such*
 760 *that any play compatible with τ has capacity bounded by c , then he has a strategy τ' in*
 761 *the 2^{c+1} -tokens explorability game such that any play compatible with τ' has its projection*
 762 *compatible with τ .*

763 We will use the notion of capacity to define the following game, using finite capacity
 764 instead of bounded to obtain a winning condition.

765 ► **Definition 30** (Capacity game). *The capacity game is played in the support arena. Given*
 766 *a play $(B_0, q_0) \xrightarrow{a_1, G_1} (B_1, q_1) \xrightarrow{a_2, G_2} (B_2, q_2) \dots$, Spoiler wins if it is a winning play in the*
 767 *support game, or if it has infinite capacity.*

768 ► **Lemma 31** ([2, Prop 3.8]). *Either Spoiler or Determiniser wins the capacity game, and*
 769 *the winner has a winning strategy with finite memory.*

770 **Proof.** Although this result talks about slightly different objects than in [2, Prop 3.8], their
 771 proof actually still stand with our definitions of capacity game and support game. The
 772 proof proceeds by building a nondeterministic Büchi automaton verifying that the capacity
 773 is infinite, determinising it into a parity automaton, and incorporating it into the arena to
 774 yield a parity game equivalent to the capacity game. The winner of this parity game has a
 775 positional strategy, which corresponds to a finite memory strategy in the capacity game. ◀

776 ► **Lemma 32** (adapted from [2, Prop 3.9]). *If Spoiler wins the capacity game, then he wins*
 777 *the k -explorability game for any k .*

778 **Proof.** Here Spoiler can simply apply the strategy for the capacity game to the explorability
 779 game, by remembering only the information that is relevant from the point of view of the
 780 capacity game (*i.e.* the supports and transfer graphs). This will simulate a realisable play of
 781 the capacity game, which has bounded capacity by Lemma 28. Since the strategy is winning
 782 in the capacity game, and this simulated play cannot have infinite capacity, Spoiler wins
 783 the underlying support game. This ensures the win for Spoiler in the explorability game:
 784 he plays a word of $L(\mathcal{A})$ as witnessed by the acceptance of \mathcal{D} , while finitely many Büchi
 785 states are witnessed by tokens of Determiniser. We use here the particular property of Büchi
 786 condition: one of the tokens follows an accepting run if and only if it occurs infinitely many
 787 times that the support set occupied by tokens contains a Büchi state. ◀

788 ► **Lemma 33** (adapted from [2, Prop 3.10]). *If Determiniser wins the capacity game using*
 789 *finite memory M , then he wins the k -explorability game for some $k \in \mathbb{N}$.*

790 **Proof.** We first prove that under these conditions, Determiniser can win the capacity game
 791 while ensuring a capacity bounded by $|M| \times |Q_{\mathcal{D}}| \times 4^{|\mathcal{Q}|}$.

792 Let us consider a winning strategy τ with memory M for Determiniser in the capacity
 793 game. We take a play $(B_0, q_0) \xrightarrow{a_1, G^1} (B_1, q_1) \xrightarrow{a_2, G^2} (B_2, q_2) \dots$ compatible with τ , and we
 794 show that its capacity is bounded by $|M| \times |Q_{\mathcal{D}}| \times 4^{|\mathcal{Q}|}$.

795 Given an accumulator $\mathcal{T} = (T_i)_{i \in \mathbb{N}}$, if there are two integers $i < j$ such that $m_i = m_j$
 796 (memory states at steps i and j), $B_i = B_j$, $q_i = q_j$ and $T_i = T_j$, then one can build a
 797 play that loops on the corresponding interval, while still being compatible with τ . This
 798 accumulator cannot have infinitely many entries, so \mathcal{T} does not have any entry in the interval
 799 $[i, j]$. As a consequence, if i and j are entry times, we have $(m_i, B_i, q_i, T_i) \neq (m_j, B_j, q_j, T_j)$,
 800 which means there can be at most $|M| \times 2^{|\mathcal{Q}|} \times |Q_{\mathcal{D}}| \times 2^{|\mathcal{Q}|} = |M| \times |Q_{\mathcal{D}}| \times 4^{|\mathcal{Q}|}$ entries in
 801 the accumulator \mathcal{T} .

802 We now know that the capacity of any play compatible with τ is bounded by $|M| \times$
 803 $|Q_{\mathcal{D}}| \times 4^{|\mathcal{Q}|}$. Take $k = 2^{1+|M| \times |Q_{\mathcal{D}}| \times 4^{|\mathcal{Q}|}}$. Lemma 29 then provides a strategy for Determiniser
 804 in the k -explorability game, that ensures that the successive supports (*i.e.* the sets of states
 805 occupied by tokens) contain Büchi states infinitely often. This means that at least one token
 806 visits Büchi states infinitely often, since there are finitely many tokens. This ensures a win
 807 for Determiniser. ◀

808 These Lemmas 32 and 33 give a way to solve the explorability problem if we can efficiently
 809 find the winner of the corresponding capacity game. Note that we could use the parity
 810 game built in the proof of Lemma 31 to solve the problem, but this would yield a doubly
 811 exponential algorithm since the parity automaton that we build in this proof is itself doubly
 812 exponential.

813 The following gives an exponential time algorithm for solving the capacity game, and
 814 therefore the explorability problem.

815 ► **Definition 34** (Leaks and separations). *If G and H are two transfer graphs, we say that G*
 816 *leaks at H if there are three states q, x, y such that $(q, y) \in G \cdot H$, $(x, y) \in H$ and $(q, x) \notin G$.*

817 *We say that G separates states r and t if there is a q such that $(q, r) \in G$ and $(q, t) \notin G$.*
 818 *The separator of G , noted $\mathbf{Sep}(G)$, is the set of all such (r, t) .*

819 *Note that in a play denoted as before, whenever $i < j < n$, we have $\mathbf{Sep}(G[i, n]) \subseteq$*
 820 *$\mathbf{Sep}(G[j, n])$.*

821 We will now define the tracking list of a play. The point of that list will be to provide an
 822 easy way to detect indices that leak infinitely often.

823 ► **Definition 35** (Tracking list). *The tracking list \mathcal{L}_n at step n is a list of transfer graphs*
 824 *$\{G[i_1, n], \dots, G[i_{k_n}, n]\}$. It is defined inductively, with \mathcal{L}_0 the empty list, and \mathcal{L}_n computed*
 825 *as follows.*

- 826 ■ *We update every $G[i, n-1]$ in \mathcal{L}_{n-1} into $G[i, n]$ by composing with G_n .*
- 827 ■ *We then add $G[n-1, n] = G_n$ at the end of the list.*
- 828 ■ *And finally we clean the list, by removing any graph with a separator identical to the*
 829 *previous one.*

830 *If for some i , $G[i, n] \in \mathcal{L}_n$ for every $n > i$, we say that i is remanent.*

831 To properly use these tracking lists, it suffices to know that the following result holds.
832 For more details we refer the to [2].

833 ► **Lemma 36** ([2, Lem 4.4]). *A play has infinite capacity if and only if there is a remanent*
834 *index that leaks infinitely often.*

835 We now define a game $\mathcal{G}_{\mathcal{A}}$ associated to \mathcal{A} , that extends the support arena using tracking
836 lists to detect infinite capacity plays. Once again, this is an adaptation from [2].

837 The **states** of $\mathcal{G}_{\mathcal{A}}$ are in $\mathcal{P}(Q) \times Q_{\mathcal{D}} \times \mathcal{G}^{\leq |Q|^2}$, where $\mathcal{G}^{\leq |Q|^2}$ is the set of lists of at most
838 $|Q|^2$ transfer graphs. Each state can be written as (B, q, L) where B is a subset of Q , q is a
839 state of \mathcal{D} , and L is a tracking list. The initial state is $(\{q_0^{\mathcal{A}}\}, q_0^{\mathcal{D}}, \varepsilon)$.

840 The **transitions** are the ones that can be written $(B, q, L) \xrightarrow{p, a, G} (B', q', L')$ with the
841 following conditions.

- 842 ■ $(B, q) \xrightarrow{a, G} (B', q')$ is a transition from the support arena.
- 843 ■ L' is obtained by updating L with G , as detailed in the definition of tracking list.
- 844 ■ Take $L = \{H_1, \dots, H_k\}$ and $L' = \{H'_1, \dots, H'_{k'}\}$. Let p' be the smallest index such that
845 $H_{p'}$ leaks at G , or $k + 1$ if there is no such index. Let p'' be the smallest index such that
846 $H'_{p''} \neq H_{p''} \cdot G$, or $k + 1$ if there is none. We then take $p = \min(2p' + 1, 2p'')$ (which
847 implies that $p \in [2, 2|Q|^2 + 1]$).

848 To choose a transition, Spoiler first chooses a letter, then Determiniser picks a transition
849 graph compatible with that letter. The rest is determined by the conditions above. This
850 creates a play that can be denoted as $(B_0, q_0, L_0) \xrightarrow{a_1, G_1, p_1} (B_1, q_1, L_1) \xrightarrow{a_2, G_2, p_2} \dots$

851 The winning condition for Spoiler goes as follows. Either the inferior limit of $(p_i)_{i>0}$ is
852 odd, or the run $(q_i)_{i \geq 0}$ is accepting while there are finitely many accepting states seen in
853 $(B_i)_{i \geq 0}$.

854 ► **Lemma 37** (adapted from [2, Thm 4.5]). *Spoiler wins $\mathcal{G}_{\mathcal{A}}$ if and only if he wins the capacity*
855 *game.*

856 **Proof.** First note that strategies in the support arena can be easily translated to $\mathcal{G}_{\mathcal{A}}$ and
857 conversely, since in both cases Spoiler only chooses letters while Determiniser picks transfer
858 graphs, and the rest is determined by these data.

859 If Spoiler has a winning strategy in $\mathcal{G}_{\mathcal{A}}$, then he can play the same strategy in the capacity
860 game. Such a play can be written as $(B_0, q_0) \xrightarrow{a_1, G_1} (B_1, q_1) \xrightarrow{a_2, G_2} \dots$, and the play of $\mathcal{G}_{\mathcal{A}}$
861 happening in the memory of Spoiler is $(B_0, q_0, L_0) \xrightarrow{a_1, G_1, p_1} (B_1, q_1, L_1) \xrightarrow{a_2, G_2, p_2} \dots$. We use
862 the notation $L_n = \{H_n^1, \dots, H_n^{k_n}\}$.

863 Since Spoiler plays according to a winning strategy in the simulated game $\mathcal{G}_{\mathcal{A}}$, at least
864 one of his winning conditions for that game hold in this play.

865 If the limit parity is $2p + 1$ for some p , then for any n large enough, H_n^p is the same as
866 H_n^{p+1} (otherwise there would be a parity less than $2p + 1$ later) and leaks infinitely often, so
867 Spoiler wins the capacity game.

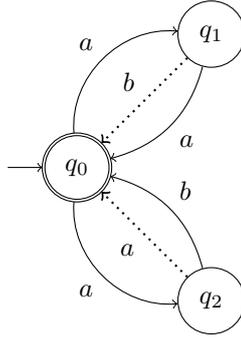
868 If the run $(q_i)_{i \geq 0}$ is accepting while there are finitely many accepting states seen in
869 $(B_i)_{i \geq 0}$, then this also ensures the win for Spoiler in the capacity game.

870 In both cases, the play is therefore won by Spoiler.

871 On the other hand, if Spoiler wins the capacity game, he can also use the same strategy
872 in $\mathcal{G}_{\mathcal{A}}$, with the same correspondence between the winning conditions.

873 ◀

874 We can finally conclude with the main result of this section:



■ **Figure 2** A co-Büchi automaton on which the projection of a play is not enough to determine the winner (the dotted lines represent co-Büchi transitions)

875 ▶ **Theorem.** *The Büchi explorability problem can be solved in EXPTIME.*

876 **Proof.** To prove this result, it is enough to prove that the game $\mathcal{G}_{\mathcal{A}}$ can be solved in
 877 exponential time in the size of \mathcal{A} , since the answer to that problem also answers the
 878 explorability of \mathcal{A} . We show that the winning condition of the game $\mathcal{G}_{\mathcal{A}}$ for Spoiler can
 879 be seen as a disjunction of parity conditions. Formally, it is of the form $\text{Parity} \vee (\text{Parity} \wedge$
 880 $\text{Co-Büchi})$. But it is straightforward to turn the second disjunct into a parity condition with
 881 twice as many priorities. Thus $\mathcal{G}_{\mathcal{A}}$ can be seen as a generalised parity game. Such games are
 882 studied in [8], which gives us an algorithm for solving $\mathcal{G}_{\mathcal{A}}$ in time $O(m^{4d}m^2) \frac{(2d)!}{d!^2}$, where d is
 883 the number of priorities and m the size of the game.

884 If we take $n = |\mathcal{A}|$, using the fact that $m = O(2^n)$, we get the complexity $O(2^{4nd+2n}) \frac{(2d)!}{d!^2}$,
 885 which can be simplified into $O(2^{4n^3+2n}(2n^2)^{n^2}) = O(2^{5n^3+2n})$ using the fact that $d = O(n^2)$.
 886 This gives us an exponential bound for the time complexity of this problem. ◀

887 ▶ **Remark 38.** We can also be interested in the number of tokens needed for Determiniser to
 888 witness explorability of an automaton. By inspecting our proof, we can see that we obtain a
 889 doubly exponential upper bound. Moreover, we can use the same construction as in [2, Prop
 890 6.3] to show that this is tight, *i.e.* some automata require a doubly exponential number of
 891 tokens to witness explorability.

892 ▶ **Remark 39.** This algorithm only works as such in the case of Büchi automata. The next
 893 step would be to adapt it to co-Büchi, with the hope that a solution for both these models
 894 might lead to one for parity automata. However, in order to use a similar method in the
 895 co-Büchi case, we would want some way to check the winning condition for a play in the
 896 explorability game using only the projection of that play in the support arena. This is not
 897 possible with the current definitions of these games: we can create plays in the explorability
 898 game with the same projection, but different winners. Take the automaton from Figure
 899 2. If we play the 2-explorability game on that automaton, Determiniser has a strategy to
 900 ensure that the support are always maximal, alternating between $\{q_0\}$ and $\{q_1, q_2\}$. However,
 901 Spoiler can either chose to always take the co-Büchi transition with the same token, or to
 902 alternate between tokens. He only wins in the second case.

903 A.3 ExpTime-hardness of the ω -explorability problem

904 This part focuses on proving Theorem 22 stating the EXPTIME-hardness of the ω -explorability
 905 problem for safety automata, which also proves the optimality of the algorithm from Sec-
 906 tion 4.2.

907 We reduce from the acceptance problem of a PSPACE alternating Turing machine. This
908 is again inspired from [2].

909 We take an alternating Turing machine $\mathcal{M} = (\Sigma_{\mathcal{M}}, Q_{\mathcal{M}}, \Delta_{\mathcal{M}}, q_0^{\mathcal{M}}, q_f^{\mathcal{M}})$ with $Q_{\mathcal{M}} =$
910 $Q_{\exists} \uplus Q_{\forall}$. It can be seen as a game between two players: existential (\exists) and universal (\forall).
911 On a given input, the game creates a run by letting \exists (resp. \forall) solve the non-determinism in
912 states from Q_{\exists} (resp. Q_{\forall}) by picking a transition from Δ . Player \exists wins if the play reaches
913 the accepting state $q_f^{\mathcal{M}}$, and w is accepted if and only if \exists has a winning strategy. We assume
914 that \mathcal{M} uses polynomial space $P(n)$ in the size n of its input, *i.e.* the winning strategies can
915 avoid configurations with tape longer than $P(n)$. We also fix an input word $w \in (\Sigma_{\mathcal{M}})^*$.

916 We will assume for simplicity that $\Sigma_{\mathcal{M}} = \{0, 1\}$ and that the machine alternates between
917 existential and universal states, starting with an existential one (meaning that $q_0 \in Q_{\exists}$ and
918 the transitions are either $Q_{\exists} \rightarrow Q_{\forall}$ or $Q_{\forall} \rightarrow Q_{\exists}$). In our reduction, this will mean that we
919 give the choice of the transition alternatively to Spoiler (playing \exists) and Determiniser (\forall).

920 We create a safety automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \perp)$ with:

921 ■ $Q = Q_{\mathcal{M}} \uplus \text{Pos} \uplus \text{Mem} \uplus \text{Trans} \uplus \{q_0, \text{store}, \top, \perp\}$ where:

922 $\text{Pos} = [1, P(n)]$

923 $\text{Mem} = \{m_{b,i} \mid b \in \{0, 1\}, i \in [1, P(n)]\}$

924 $\text{Trans} = \{E\} \cup \{A_t \mid t \in \Delta_{\mathcal{M}}\}$
925

926 ■ $\Sigma = \{a_{t,p} \mid t \in \Delta_{\mathcal{M}} \text{ and } p \in [1, P(n)]\} \uplus \{\text{init}, \text{end}, \text{restart}, \text{win}\} \uplus \{\text{check}_q \mid q \in$
927 $Q_{\mathcal{M}}\} \uplus \{\text{check}_{b,i} \mid (b, i) \in \{0, 1\} \times [1, P(n)]\}$.

928 ■ \perp is a rejecting sink state: a run is accepting if and only if it never reaches this state.

929 Let us give the intuition for the role of each state of \mathcal{A} . First the states in $Q_{\mathcal{M}}$, Pos and
930 Mem are used to keep track of the configuration of \mathcal{M} , as described in Lemma 40. Those
931 in Trans are used to simulate the choices of \exists and \forall (played by Spoiler and Determiniser
932 respectively). The state **store** keeps tokens safe for the remaining of a run when Spoiler
933 decides to ignore their transition choice. The sinks \top and \perp are respectively the one Spoiler
934 must avoid at all cost, and the one in which he wants to send every token eventually.

935 We now define the transitions in Δ . The states \top and \perp are both sinks (\top accepting and
936 \perp rejecting). We then describe all transitions labelled by the letter $a_{t,p}$ with $p \in \text{Pos}$ and
937 $t = (q, q', b, b', d) \in \Delta_{\mathcal{M}}$, where q and q' are the starting and destination states of t , while b
938 and b' are the letters read and written at the current head position, and $d \in \{L, R\}$ is the
939 direction taken by the head. These transitions are:

940 ■ $q \rightarrow q'$.

941 ■ $p \rightarrow p'$ with $p' = p + 1$ if $d = R$, or $p - 1$ if $d = L$. It goes to \top if $p' \notin [1, P(n)]$.

942 ■ $m_{b,p} \rightarrow m_{b',p}$, and $m_{b'',p''} \rightarrow m_{b',p''}$ for any b'' and any $p'' \neq p$.

943 ■ $E \rightarrow A_{t'}$ for any transition t' .

944 ■ $A_t \rightarrow E$.

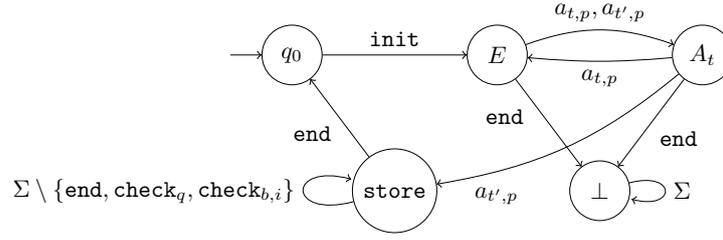
945 ■ $q'' \rightarrow \top$ for any $q'' \neq q$.

946 ■ $m_{1-b,p} \rightarrow \top$ ($1 - b$ is the boolean negation of b).

947 ■ $p' \rightarrow \top$ for any $p' \neq p$.

948 ■ $A_{t'} \rightarrow \text{store}$ for any transitions $t' \neq t$.

949 The first three bullet points manage the evolution of the configuration of \mathcal{M} . The next two
950 deal with the alternation between players, and the next three punish Spoiler if the transition
951 is invalid (the **check** letters will handle the case where Determiniser is the one giving an
952 invalid transition). The last one saves the tokens that are not chosen for the transition.



■ **Figure 3** Gadget for simulating the choice of \forall in the alternation (transitions labelled by **check** are not represented, and t' represents any transition different from t).

953 The other letters give the following transitions.

- 954 ■ **init** goes from q_0 to the states E , q_0^M , and $1 \in \text{Pos}$, and also to the states $m_{b,i}$ corresponding to the initial content of the tape, *i.e.* all $m_{b,i}$ such that b is the i -th letter of w (or 0 if $i > |w|$).
- 956 ■ **end** labels transitions from any non accepting state of \mathcal{M} to \top , from **store** to q_0 , and from any other state to \perp .
- 958 ■ **check_q** creates a transition from A_t to \perp for any $t \in \Delta$ starting from q . It also creates a transition from q to \top . Any other state is sent back to q_0 . Intuitively, playing that letter means that q is not the current state and that any transition starting from q is invalid.
- 959 ■ **check_{b,i}** creates a transition from A_t to \perp for any $t \in \Delta$ reading b on the tape. It also creates transitions from any $j \in \text{Pos} \setminus \{i\}$ and from $m_{b,i}$ to \top . Any other state is sent to q_0 . Intuitively, playing that letter means that the current head position is i , and that its content is not b , so any transition reading b is invalid.

966 To summarize, the states of \mathcal{A} can be seen as two blocks, apart from q_0 , \top and \perp : those dealing with the configuration of \mathcal{M} ($Q_{\mathcal{M}}$, Pos and Mem), and those from the gadget of Figure 3 which deal with the alternation and non deterministic choices.

968 The following result provides tools to manipulate the relation between \mathcal{A} and \mathcal{M} .

970 ► **Lemma 40.** *Let us consider a play of the ω -explorability game on \mathcal{A} , that we stop at some point. Suppose that the letters $a_{t,p}$ played since the last **init** are $a_{t_1,p_1}, \dots, a_{t_k,p_k}$. If \top is not reachable from q_0 with this sequence, then we can define a run ρ of \mathcal{M} on w taking the sequence of transitions t_1, \dots, t_k . The following implications hold:*

Token present in	implies that at the end of ρ
$q \in Q_{\mathcal{M}}$	the current state is q
$p \in \text{Pos}$	the head is in position p
$m_{b,i} \in \text{Mem}$	the tape contains b at position i
E	it is the turn of \exists
A_t	it is the turn of \forall

974
975 **Proof.** These results are obtained by straightforward induction from the definitions. The unreachability of \top is used to ensure that only valid transitions are played. ◀

977 We will now prove that \mathcal{A} is ω -explorable if and only if the Turing machine \mathcal{M} rejects the word w . Let us first assume that $w \in \mathcal{L}(\mathcal{M})$. There is a winning strategy σ_{\exists} for \exists in the alternating Turing machine game, and Spoiler will use that strategy in the explorability game to win against ω tokens. He will consider that the tokens are labelled by integers, and always target the smallest one that is not already in \perp . He proceeds as follows.

- 982 ■ Spoiler plays **init** from a position where every token is either in q_0 or \perp . We can assume
 983 from here that Determiniser sends token to each possible state, and just add imaginary
 984 tokens if he does not. Additionally, if the target token does not go to E , then Spoiler
 985 creates an imaginary target token in E that will play only valid transitions (we will
 986 describe what this means later). Its purpose is to ensure that we actually reach an
 987 accepting state of \mathcal{M} to destroy the real target token.
- 988 ■ When there are tokens in E , Spoiler plays letters according to σ_{\exists} . More formally,
 989 if the letters played since **init** are $a_{t_1, p_1} \dots a_{t_i, p_i}$, then Spoiler plays $a_{t_{i+1}, p_{i+1}}$ where
 990 $t_{i+1} = \sigma_{\exists}(t_1, \dots, t_i)$ and $p_{i+1} = p_i + 1$ or $p_i - 1$ depending on the head movement in t_i .
- 991 ■ After such a play, Determiniser can move tokens to any state A_t . If there are more than
 992 one occupied state, Spoiler picks the one containing the current target token (possibly
 993 imaginary).
- 994 ■ If that state corresponds to an invalid transition (wrong starting state or wrong tape
 995 content at the current head position), then Spoiler plays the corresponding **check**
 996 letter. Formally, if the target token (not the imaginary one, since Spoiler can avoid
 997 invalid transitions for that one) is in \mathcal{A}_t , Spoiler plays **check_q** if the starting state q of
 998 t does not match the current state of the tape (given by Lemma 40), or **check_{b,i}** if the
 999 current head position is i and does not contain b . In both cases, the target token is
 1000 sent to \perp with no other token reaching \top (by Lemma 40). This sends us back to the
 1001 first step, but with an updated target.
 - 1002 ■ If the state instead corresponds to a valid transition, then Spoiler can play the
 1003 corresponding $a_{t,p}$, where p is the current head position (again, given by Lemma 40),
 1004 then go back to the previous step (where there are tokens in E).
- 1005 ■ If no invalid transition is reached, the run eventually gets to an accepting state of \mathcal{M}
 1006 because σ_{\exists} is winning. This corresponds to a stage where Spoiler can safely play **end** to
 1007 get rid of the target token along with all tokens outside of **store**, by sending them to \perp
 1008 (the only reason not to play **end** would be the existence of tokens in non accepting states
 1009 of $Q_{\mathcal{M}}$). This sends us back to the first step, but with an updated target.

1010 This strategy guarantees that after k runs, at least the first k tokens are in state \perp , and
 1011 therefore cannot witness an accepting run. We also know that the final word is accepted by
 1012 \mathcal{A} , because an accepting run can be created by going to the state **store** as soon as possible
 1013 in each factor corresponding to a run of \mathcal{M} .

1014 Conversely, if there is a winning strategy σ_{\forall} for the universal player in the alternation
 1015 game on $\mathcal{M}(w)$, then we can build a winning strategy for Determiniser in the ω -explorability
 1016 game. This strategy is more straightforward than the previous one, as we can focus on the
 1017 tokens sent to E (while still populating each state when **init** is played, but these other
 1018 tokens follow a deterministic path until the next **init**).

1019 Determiniser will initially chooses a specific token, called leader. He then sends ω tokens
 1020 to every reachable state when Spoiler plays **init**, with the leader going to E . Determiniser
 1021 then moves the tokens in the leader's state according to σ_{\forall} . Spoiler cannot send the leader to
 1022 \perp , since the only way to do that would be using the letter **end**, but this would immediately
 1023 ensure the win for Spoiler, as there will always be some token in non-accepting states of \mathcal{M}
 1024 (because σ_{\forall} is winning), and those tokens would be sent to \top upon playing **end**. This means
 1025 that Spoiler has no way to send the leader to \perp without losing the game, and therefore that
 1026 Determiniser wins.

1027 Note that with that strategy, Spoiler can still safely send some tokens to \perp by playing
 1028 the wrong transition, which sends the tokens following the leader to **store**, then some well

23:26 Explorable automata

1029 chosen check letter to send the remaining ones to \perp . However Determiniser will start the
1030 next run with still ω tokens, including the leader. This is why the choice of a specific leader
1031 is important, as it can never be safely sent to \perp .

1032 This proves that the automaton \mathcal{A} created from \mathcal{M} and w (using polynomial time) is
1033 ω -explorable if and only if \mathcal{M} rejects w . This completes the proof since the acceptance
1034 problem is EXPTIME-hard for alternating Turing machines using polynomial space.