On the size of good-for-games Rabin automata and its link with the memory in Muller games

Antonio Casares, Thomas Colcombet, Karoliina Lehtinen

6 July 2022

ICALP 2022
Two main tools used in the verification and synthesis of non-terminating reactive systems:

- Automata over infinite words.
- Two-players infinite duration games over graphs.
Main Contribution: Automata and Strategy Complexity

Good-For-Games Automata

- Automata with a restricted amount of non-determinism.
- Applications in synthesis.

Memories for Games

- Structures implementing strategies.
- Measure of the complexity of strategies.
Muller Games and Memory Structures
Infinite Duration Games

\[ \mathcal{G} = (V = V_{\text{Eve}} \sqcup V_{\text{Adam}}, E, v_0) \]

Players move a token in turns producing an infinite word \( w \in C^\omega \).
Infinite Duration Games

Players move a token in turns producing an infinite word $w \in C^\omega$.

Output =
Infinite Duration Games

Players move a token in turns producing an infinite word $w \in C^\omega$.

Output $= a$

$C = \{a, b, c\}$
Infinite Duration Games

Players move a token in turns producing an infinite word $w \in C^\omega$.

Output $= ab$
Infinite Duration Games

Players move a token in turns producing an infinite word $w \in C^\omega$.

Output $= abb \ldots$
The winning condition is given by a language

\[ L \subseteq C^\omega. \]

Eve wins if the sequence \( w \in C^\omega \) produced belongs to \( L \).
The winning condition is given by a language

\[ L \subseteq C^\omega. \]

Eve wins if the sequence \( w \in C^\omega \) produced belongs to \( L \).

We study a restricted class of languages: Muller languages.
Muller Languages

Muller language

Language described by boolean combination of

“Letter $x$ appears infinitely often”.

“Letter $y$ appears only finitely often”.

Examples

$L_1 = \{ w \in C^\omega \mid w \text{ contains ‘}a\text{’ and ‘}b\text{’ infinitely often}\}.$

$L_2 = \{ w \in C^\omega \mid \text{If $w$ contains ‘}a\text{’ infinitely often, then ‘}c\text{’ appears finitely many times}\}.$
Strategies Might Require Memory

$L = \text{See ‘a’ and ‘b’ infinitely often.}$

Eve can force a victory, but she needs to remember previous moves.
Strategies Might Require Memory

$L = \text{See ‘a’ and ‘b’ infinitely often.}$

Eve can force a victory, but she needs to remember previous moves.

→ We use memory structures.
Memory Structures

- Set of states $M$.
- Update function.
- next-move: $M \times V_{Eve} \to E$, gives a strategy.

$L = \text{See 'a' and 'c' infinitely often.}$
Memory Structures

- Set of states $M$.
- Update function.
- next-move: $M \times V_{\text{Eve}} \rightarrow E$, gives a strategy.

$L = \text{See 'a' and 'c' infinitely often.}$

Output $= a$
Memory Structures

- Set of states $M$.
- Update function.
- next-move: $M \times V_{Eve} \rightarrow E$, gives a strategy.

$L = $ See ‘$a$’ and ‘$c$’ infinitely often.

Output $= ab$
Memory Structures

- Set of states $M$.
- Update function.
- next-move: $M \times V_{Eve} \rightarrow E$, gives a strategy.

Output = $abc \ldots$

$L = \text{See 'a' and 'c' infinitely often.}$
Memory Structures as a Measure of the Strategy Complexity

Number of states of a memory structure $\sim$ Complexity of the strategy
Optimal Size for Memory Structures

Memory Requirements

For $L$ a Muller language, $\text{mem}(L)$ is the minimal $n \in \mathbb{N}$ such that if Eve wins an $L$-game, she can win it using a memory with $n$ states.

- Always finite (Gurevich, Harrington 1982).
- Characterised by Dziembowsk and Jurdzinski and Walukiewicz in 1997.
Question 1

Give a structural description of optimal memories in \( L \)-games, for \( L \) a Muller language.
Good-For-Games Rabin Automata
Automata Over Infinite Words

Automaton $A$, $\Sigma = \{a, b\}$.

Input $= abababbbbab \cdots \in \Sigma^\omega \rightarrow$ Infinite runs over the automaton.
Automata Over Infinite Words

Automaton $\mathcal{A}$, $\Sigma = \{a, b\}$.

Input $= abababbbbaab \cdots \in \Sigma^\omega \rightarrow$ Infinite runs over the automaton.

→ Different possibilities to define which runs will be accepting (Büchi, parity, Rabin...).
Büchi condition: We accept if we visit infinitely often a transition.
**co-Büchi condition:** We accept if we visit only finitely often a $\times$-transition.
Rabin condition: A general acceptance condition expressing fairness properties that:

- Encompass Büchi, co-Büchi and parity.
- Appears naturally in verification problems (determinisation of automata, decidability of MSO over trees).
$\mathcal{A}$ a non-deterministic automaton recognizing $L \subseteq \Sigma^\omega$.

$\mathcal{A}$ is \textit{good-for-games} if there exists a strategy $\sigma$ resolving its non-determinism such that:

$$w \in L \iff \text{The run over } w \text{ obtained following } \sigma \text{ is accepting.}$$
Good-For-Games (GFG) Automata

\( \mathcal{A} \) a non-deterministic automaton recognizing \( L \subseteq \Sigma^\omega \).

\( \mathcal{A} \) is \textit{good-for-games} if there exists a strategy \( \sigma \) resolving its non-determinism such that:

\[ w \in L \iff \text{The run over } w \text{ obtained following } \sigma \text{ is accepting.} \]

"Guessing" is not allowed
Examples GFG Automata

Deterministic $\Rightarrow$ Good-For-Games

But they are not the only ones!
Non-Example GFG Automata

\[ w \in L(A) \iff \text{Eventually, } w \text{ only contains letter \textquote{a}.} \]
Example GFG Automata

GFG but not deterministic!

\[ w \in L(A) \iff w \text{ does not contain both } 'a' \text{ and } 'c' \text{ infinitely often.} \]
Application of GFG Automata

- Intermediate model between deterministic and non-deterministic.
- Share some good properties with deterministic automata, and can be used instead for some applications.
- They can be smaller than deterministic ones.
Application of GFG Automata

- Intermediate model between deterministic and non-deterministic.
- Share some good properties with deterministic automata, and can be used instead for some applications.
- They can be smaller than deterministic ones.

1. Deterministic automata are much bigger than non-deterministic ones.
2. Determinization is very costly for automata over infinite words.
Application of GFG Automata

- Intermediate model between deterministic and non-deterministic.
- Share some good properties with deterministic automata, and can be used instead for some applications.
- They can be smaller than deterministic ones.

Question 2
When are GFG automata more succinct than deterministic ones?

Question 3
Provide general tools for constructing GFG automata.
Main Results
Theorem

For every Muller language $L \subseteq C^\omega$, the following quantities coincide:

- The size of a minimal GFG-Rabin automaton recognising $L$.
- $\text{mem}(L)$: the optimal memory for the winning condition $L$.

→ Unexpected and fundamental role of GFG automata!
Contribution 2: Construction of Minimal GFG-Rabin Automata

Good-For-Games Automata \(\leftrightarrow\) Memories for Games

Upper bound for GFG-automata \(\rightarrow\) Explicit construction of a minimal GFG-Rabin automaton for a Muller language.

Remark: Constructing such minimal deterministic Rabin automata is NP-complete.
Contribution 3: Succinctness of GFG-Rabin Automata

Theorem

*Good-for-games Rabin automata recognising Muller languages can be exponentially more succinct than deterministic ones.*
Theorem

Good-for-games Rabin automata recognising Muller languages can be exponentially more succinct than deterministic ones.

Proof idea:

Lower bounds for the chromatic number of some graphs. $\implies$ Lower bounds for deterministic Rabin automata.
Conclusions
Conclusions

Together with previous work (CSL ’22) we obtain a complete characterisation of different memories for Muller languages:

\[
\begin{array}{c}
\text{General Memory Structure for } L = \text{ GFG Rabin Automaton for } L \\
\text{Exponential gap on the size}
\end{array}
\]

\[
\begin{array}{c}
\text{Chromatic Memory Structure for } L = \text{ Deterministic Rabin Automaton for } L \\
\end{array}
\]

Thanks for your attention!
Conclusions

Together with previous work (CSL ’22) we obtain a complete characterisation of different memories for Muller languages:

Thanks for your attention!