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

Computing over infinite words

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

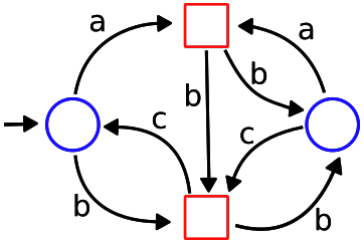
- Automata with a restricted amount of non-determinism.
- Applications in synthesis.

Memories for Games

- Structures implementing strategies.
- Measure of the complexity of strategies.

Muller Games and Memory Structures

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

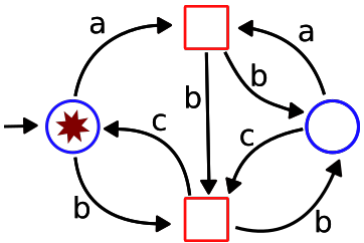


Eve

Adam

$C = \{a, b, c\}$

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



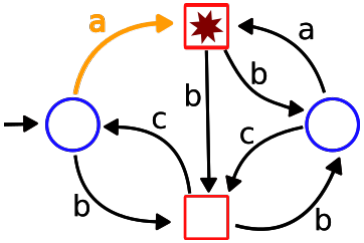
Eve

Adam

$C = \{a, b, c\}$

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

$$\text{Output} =$$



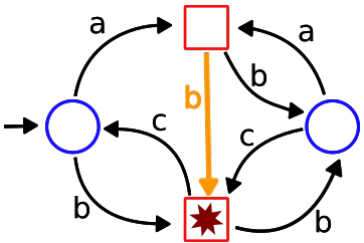
Eve

Adam

$C = \{a, b, c\}$

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

$$\text{Output} = a$$



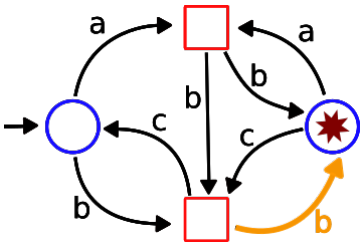
Eve

Adam

$C = \{a, b, c\}$

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

$$\text{Output} = ab$$



Eve

Adam

$C = \{a, b, c\}$

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

$$\text{Output} = abb\ldots$$

Winning Condition

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.

Winning Condition

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".

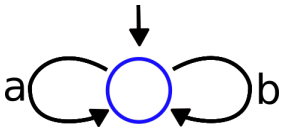
"Letter y appears only finitely often".

Examples

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

$L_2 = \{w \in C^\omega \mid \text{If } w \text{ contains '}a\text{' infinitely often, then '}c\text{' appears finitely many times}\}$.

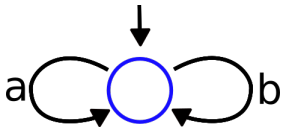
Strategies Might Require Memory



L = See ‘a’ and ‘b’ infinitely often.

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

Strategies Might Require Memory



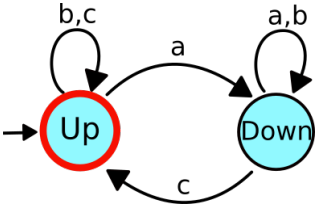
L = See ‘a’ and ‘b’ infinitely often.

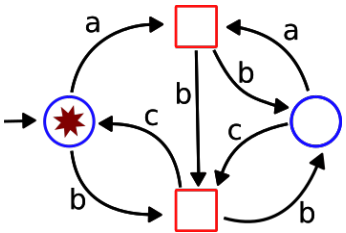
Eve can force a victory, but she needs to remember previous moves.

➔ We use **memory structures**.

Memory Structures

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



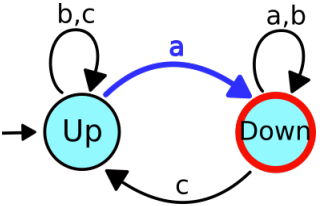


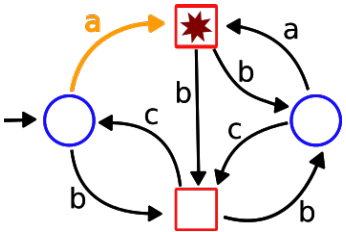
$\mathrm{Output} =$

$L =$ See 'a' and 'c' infinitely often.

Memory Structures

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



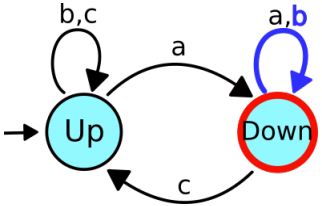


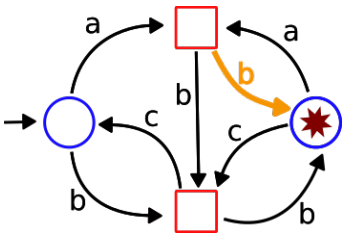
$\mathrm{Output} = a$

$L =$ See 'a' and 'c' infinitely often.

Memory Structures

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



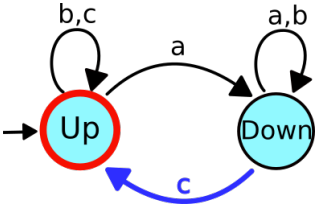


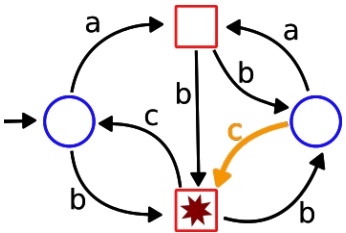
$\mathrm{Output} = ab$

$L =$ See 'a' and 'c' infinitely often.

Memory Structures

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





$\mathrm{Output} = abc\ldots$

$L =$ 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, $\mathrm{mem}(L)$ 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.

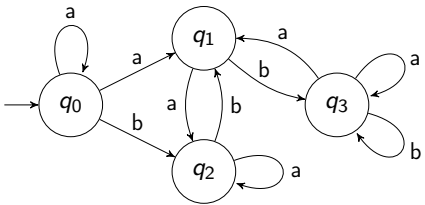
- ➔ Always finite (Gurevich, Harrington 1982).
- ➔ Characterised by Dziembowski and Jurdzinski and Walukiewicz in 1997.

Structural Description of Optimal Memory Structures

Question 1

Give a structural description of optimal memories in L-games,
for L a Muller language.

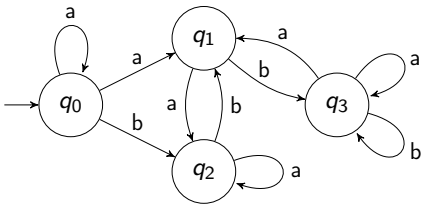
Good-For-Games Rabin Automata



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

$Input = ababababbbbbab \cdots \in \Sigma^{\omega} \quad \longrightarrow \quad$ Infinite runs over the automaton.

Automata Over Infinite Words

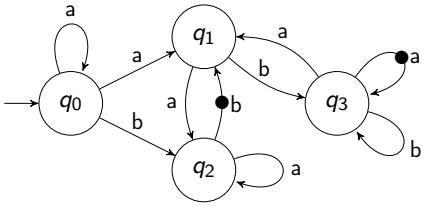


Automaton $\mathcal{A}$, $\Sigma = \{a, b\}$.

$Input = ababababbbbbab \cdots \in \Sigma^{\omega} \quad \longrightarrow \quad$ Infinite runs over the automaton.

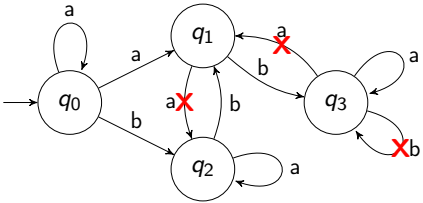
➔ Different possibilities to define which runs will be accepting (Büchi, parity, **Rabin**...).

Example: Büchi Conditions



Büchi condition: We accept if we visit infinitely often a ●→ transition.

Example: co-Büchi Conditions



co-Büchi condition: We accept if we visit only finitely often a X-transition.

Rabin Conditions

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).

Good-For-Games (GFG) Automata

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

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

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

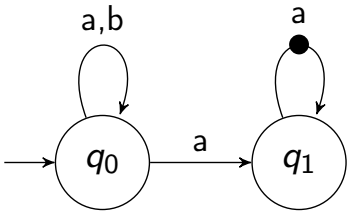
"Guessing" is not allowed

Deterministic $\Longrightarrow$ Good-For-Games

But they are not the only ones!

Non-Example GFG Automata

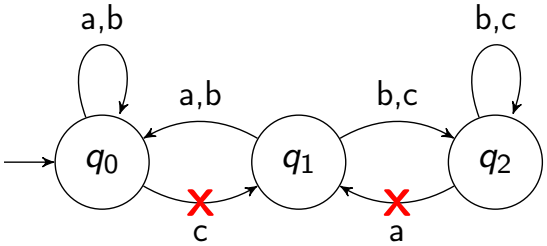
NOT GFG!



$w \in L(\mathcal{A}) \Leftrightarrow$ Eventually, w only contains letter ‘a’.

Example GFG Automata

GFG but not deterministic!



$w \in L(\mathcal{A}) \Leftrightarrow$ w does not contain both ‘a’ and ‘c’ 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

Contribution 1: Correspondence Memory $\leftrightarrow$ GFG-Automata

Theorem

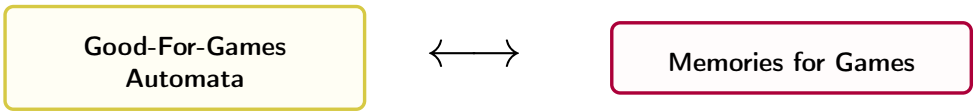
For every Muller language $L \subseteq C^\omega$, the following quantities coincide:

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

Good-For-Games Automata $\longleftrightarrow$ **Memories for Games**

➔ Unexpected and fundamental role of GFG automata!

Contribution 2: Construction of Minimal GFG-Rabin Automata



Upper bound for GFG-automata ➔ **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.*

Contribution 3: Succinctness of GFG-Rabin Automata

Theorem

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

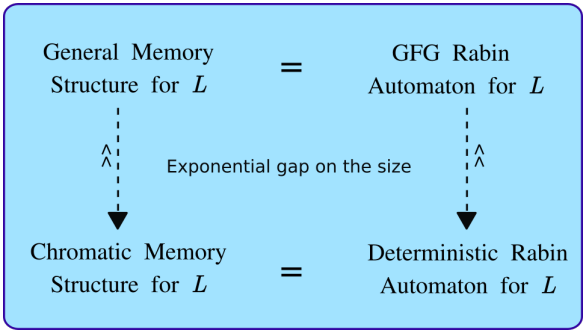
Proof idea:

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

Conclusions

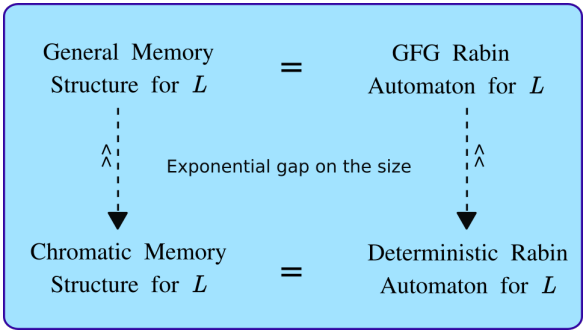
Conclusions

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



Conclusions

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



Thanks for your attention!