On a Correspondence Between Memory Structures and Rabin Automata

Antonio Casares

Séminaire de l’équipe LX

21 Octobre 2021

Based in work with Thomas Colcombet and Karoliina Lehtinen.
1. Different Memory Structures For Muller Games
2. Deterministic Rabin Automata and Independent Memory
3. Good-For-Games Rabin Automata and General Memory
1 Different Memory Structures For Muller Games

2 Deterministic Rabin Automata and Independent Memory

3 Good-For-Games Rabin Automata and General Memory
**Infinite Duration Games**

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

Players move a token in turns producing an infinite word \( w \in \{a, b, c\}^\omega \).
**Muller Conditions**

Let $\mathcal{C}$ be the set of colours (for example, $\mathcal{C} = \{a, b, c\}$).

**Muller condition:** Family of subsets of colours $\mathcal{F} \subseteq 2^\mathcal{C}$.

**Example**

Produce both “$a$” and “$c$” infinitely often:

$$\mathcal{F} = \{ \{a, c\}, \{a, b, c\} \}.$$
Muller Games

Figure: $\mathcal{F}$-game

Muller condition:

$$\mathcal{F} = \{ \{a, c\}, \{a, b, c\}\}.$$  

Eve wins if the produced word $w \in C^\omega$ verifies

$$\text{Inf}(w) \in \mathcal{F}.$$
Muller Games Might Require Memory

\[ F = \{ \{ a, b \} \}. \]

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

→ We use memory structures.
General Memory Structures

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

$\mathcal{F} = \{ \{a, c\}, \{a, b, c\} \}$. 
Different Memory Structures For Muller Games

General Memory Structures

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

$F = \{ \{a, c\}, \{a, b, c\} \}$. 
Different Memory Structures For Muller Games

General Memory Structures

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

$\mathcal{F} = \{ \{a, c\}, \{a, b, c\} \}$. 
General Memory Structures

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

$$F = \{ \{a, c\}, \{a, b, c\} \}.$$
Different Memory Structures For Muller Games

**General Memory Structures**

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

\[ \mathcal{F} = \{ \{a, c\}, \{a, b, c\} \} \]
General Memory Structures

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

$F = \{ \{a, c\}, \{a, b, c\} \}$. 
General Memory Structures

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

$$\mathcal{F} = \{\{a, c\}, \{a, b, c\}\}.$$
Chromatic Memory Structures

- Set of states \( M \).
- \( \mu: M \times C \rightarrow M \), update function.
- next-move: \( M \times V_{Eve} \rightarrow E \), gives a strategy.

\[ F = \{ \{a, c\}, \{a, b, c\} \}. \]
Different Memory Structures For Muller Games

**Arena-Independent Memory Structures**

We fix a Muller condition $\mathcal{F}$.

- Set of states $M$.
- $\mu: M \times \mathcal{C} \rightarrow M$, update function.

Memory structure $\mathcal{M} = (M, m_0, \mu)$.

The (chromatic) memory $\mathcal{M}$ is **arena-independent** if for every $\mathcal{F}$-game $\mathcal{G}$ won by Eve, there is a winning strategy given by some function

$$\text{next-move}_\mathcal{G}: M \times V_{\text{Eve}} \rightarrow E$$
We fix a set of colours $C$ and a Muller condition $\mathcal{F} \subseteq 2^C$.

<table>
<thead>
<tr>
<th>General Memory Requirements</th>
<th>$= \text{mem}_{gen}(\mathcal{F})$</th>
<th>Minimal $n$ such that for any $\mathcal{F}$-game won by Eve, she can win it using a general memory of size $n$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromatic Memory Requirements</td>
<td>$= \text{mem}_{chr}(\mathcal{F})$</td>
<td>Minimal $n$ such that for any $\mathcal{F}$-game won by Eve, she can win it using a chromatic memory of size $n$.</td>
</tr>
<tr>
<td>Arena-Independent Memory Requirements</td>
<td>$= \text{mem}_{ind}(\mathcal{F})$</td>
<td>Minimal size of an arena-independent memory for $\mathcal{F}$.</td>
</tr>
</tbody>
</table>
### General Memory Requirements

$$\text{General Memory Requirements } = \text{mem}_{\text{gen}}(\mathcal{F})$$

Minimal $n$ such that for any $\mathcal{F}$-game won by Eve, she can win it using a general memory of size $n$.

### Chromatic Memory Requirements

$$\text{Chromatic Memory Requirements } = \text{mem}_{\text{chr}}(\mathcal{F})$$

Minimal $n$ such that for any $\mathcal{F}$-game won by Eve, she can win it using a chromatic memory of size $n$.

### Arena-Independent Memory Requirements

$$\text{Arena-Independent Memory Requirements } = \text{mem}_{\text{ind}}(\mathcal{F})$$

Minimal size of an arena-independent memory for $\mathcal{F}$.

$$\text{mem}_{\text{gen}}(\mathcal{F}) \leq \text{mem}_{\text{chr}}(\mathcal{F}) \leq \text{mem}_{\text{ind}}(\mathcal{F})$$
<table>
<thead>
<tr>
<th>Memory Structure</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Memory</td>
<td>$\text{mem}_{gen}(\mathcal{F})$</td>
<td>Minimal $n$ such that for any $\mathcal{F}$-game won by Eve, she can win it using a general memory of size $n$.</td>
</tr>
<tr>
<td>Chromatic Memory</td>
<td>$\text{mem}_{chr}(\mathcal{F})$</td>
<td>Minimal $n$ such that for any $\mathcal{F}$-game won by Eve, she can win it using a chromatic memory of size $n$.</td>
</tr>
<tr>
<td>Arena-Independent Memory</td>
<td>$\text{mem}_{ind}(\mathcal{F})$</td>
<td>Minimal size of an arena-independent memory for $\mathcal{F}$.</td>
</tr>
</tbody>
</table>

$\text{mem}_{gen}(\mathcal{F}) < \text{mem}_{chr}(\mathcal{F}) \quad \text{Kopczynski}^{'08} \quad \text{mem}_{ind}(\mathcal{F})$
Contributions

$\mathcal{C}$ set of colours, $\mathcal{F} \subseteq 2^\mathcal{C}$.

\[
\text{mem}_{\text{gen}}(\mathcal{F}) = \text{Size of a minimal Good-For-Games Rabin automaton recognizing } \mathcal{F}.
\]

\[
\text{mem}_{\text{chr}}(\mathcal{F}) = \text{mem}_{\text{ind}}(\mathcal{F}) = \text{Size of a minimal deterministic Rabin automaton recognizing } \mathcal{F}.
\]

$\rightarrow$ The gap between $\text{mem}_{\text{gen}}(\mathcal{F})$ and $\text{mem}_{\text{chr}}(\mathcal{F})$ can be exponential in $|\mathcal{C}|$. 
Index

1 Different Memory Structures For Muller Games

2 Deterministic Rabin Automata and Independent Memory

3 Good-For-Games Rabin Automata and General Memory
Rabin Conditions

Class of Muller conditions $\mathcal{F} \subseteq 2^C$ defined by:

A set of Rabin pairs: $(P_1, P_2, P_3, \ldots, P_k)$

Each colour in $C$ triggers one light for each Rabin pair $P_i$:

We accept a word $w \in C^\omega$ if some $P_i$ produces infinitely often green and only finitely many times red.
Rabin Conditions

Example

“See at most two colours infinitely often” is a Rabin condition:

\[ \mathcal{F} = \{ \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\} \}. \]
Rabin Conditions

**Theorem (Klarlund’94, Zielonka’98)**

*Rabin conditions are exactly the family of Muller conditions that are positionally determined* (that is, if Eve wins a Rabin game, she can win it using a memoryless strategy).
Rabin Automata

Input alphabet → $C = \{a, b, c\}$.

Output alphabet → $C' = \{\alpha, \beta\}$.

Acceptance condition → Rabin condition over $C'$: $F' \subseteq 2^{C'}$.

The automaton $\mathcal{A}$ recognizes a Muller condition $\mathcal{F}$ over $C$ if

$$\mathcal{L}(\mathcal{A}) = \{w \in C^\omega : \text{Inf}(w) \in \mathcal{F}\}.$$
### Rabin Automata are Arena-Independent Memories

| \( G \rightarrow \) | \( F \)-game |
| \( A \rightarrow \) | Automaton recognizing \( F \), Acceptance condition \( F' \). |

\( G \times A \)

Product game

Winning condition \( F' \).
Rabin Automata are Arena-Independent Memories

\[ G \rightarrow \ \mathcal{F}\text{-game} \]
\[ A \rightarrow \ \text{Automaton recognizing } \mathcal{F}, \text{Acceptance condition } \mathcal{F}'. \]

\[ \{ \text{Product game} \}
\[ \text{Eve wins } G \iff \text{Eve wins } G \times A. \]

Lemma (Folklore)

*If \( A \) is deterministic,*

\[ \text{Eve wins } G \iff \text{Eve wins } G \times A. \]
Rabin Automata are Arena-Independent Memories

\[ G \rightarrow \mathcal{F}\text{-game} \]
\[ A \rightarrow \text{Automaton recognizing } \mathcal{F}, \]
\[ \text{Acceptance condition } \mathcal{F}'. \]

\{ \begin{align*}
\text{Product game} \\
G \times A \\
\text{Winning condition } \mathcal{F}'.
\end{align*} \}

Lemma (Folklore)

If \( A \) is deterministic,

\[ \text{Eve wins } G \iff \text{Eve wins } G \times A. \]

We can transform a positional strategy in \( G \times A \) into a next-move function

\[ \text{next-move}_G : A \times V_{\text{Eve}} \rightarrow E. \]

Corollary (Folklore)

If \( A \) is a deterministic Rabin automata recognizing a Muller condition \( \mathcal{F} \),
then \( A \) is an arena-independent memory for \( \mathcal{F} \).
Theorem (C. ’21)

If $M$ is an arena-independent memory for $\mathcal{F}$, then we can define a Rabin condition on top of it so that it becomes a deterministic Rabin automaton recognizing $\mathcal{F}$.

Corollary

$$\text{mem}_{chr}(\mathcal{F}) = \text{Size of a minimal deterministic Rabin automaton for } \mathcal{F}.$$
Index

1. Different Memory Structures For Muller Games

2. Deterministic Rabin Automata and Independent Memory

3. Good-For-Games Rabin Automata and General Memory
Good-For-Games Automata

\( \mathcal{A} \) a \textbf{Non-Det} automaton recognizing \( \mathcal{F} \), using acceptance condition \( \mathcal{F}' \).

If \( \mathcal{G} \) is an \( \mathcal{F} \)-game, we can also build the product game \( \mathcal{G} \times \mathcal{A} \), but the winner is not necessarily preserved!
**Good-For-Games Automata**

\(\mathcal{A}\) a **Non-Det** automaton recognizing \(\mathcal{F}\), using acceptance condition \(\mathcal{F}'\).

If \(\mathcal{G}\) is an \(\mathcal{F}\)-game, we can also build the product game \(\mathcal{G} \times \mathcal{A}\), but the winner is not necessarily preserved!

**Definition**

\(\mathcal{A}\) is **Good-For-Games (GFG)** if for every \(\mathcal{F}\)-game \(\mathcal{G}\),

\[
\text{Eve wins } \mathcal{G} \iff \text{Eve wins } \mathcal{G} \times \mathcal{A}.
\]

(Remark: in \(\mathcal{G} \times \mathcal{A}\) Eve controls vertices corresponding to transitions in \(\mathcal{A}\).)
Good-For-Games Automata

$\mathcal{A}$ a **Non-Det** automaton recognizing $\mathcal{F}$, using acceptance condition $\mathcal{F}'$. If $\mathcal{G}$ is an $\mathcal{F}$-game, we can also build the product game $\mathcal{G} \times \mathcal{A}$, but the winner is not necessarily preserved!

**Definition**

$\mathcal{A}$ is **Good-For-Games** (GFG) if for every $\mathcal{F}$-game $\mathcal{G}$,

$$\text{Eve wins } \mathcal{G} \iff \text{Eve wins } \mathcal{G} \times \mathcal{A}.$$ 

(Remark: in $\mathcal{G} \times \mathcal{A}$ Eve controls vertices corresponding to transitions in $\mathcal{A}$.)

**Facts**

- Deterministic automata are Good-For-Games.
- There are Good-For-Games automata that are not deterministic.
If $\mathcal{A}$ is a GFG-Rabin automaton recognizing $\mathcal{F}$, we can do as before:

We build $\mathcal{G} \times \mathcal{A}$, where Eve has a positional strategy. We can transform this strategy in a next-move function

$$\text{next-move}_G : \mathcal{A} \times V_{\text{Eve}} \rightarrow E.$$ 

So $\mathcal{A}$ provides a(n) ? memory structure.
If $\mathcal{A}$ is a GFG-Rabin automaton recognizing $\mathcal{F}$, we can do as before:

We build $\mathcal{G} \times \mathcal{A}$, where Eve has a positional strategy. We can transform this strategy in a next-move function

$$\text{next-move}_G : \mathcal{A} \times V_{Eve} \rightarrow E.$$ 

So $\mathcal{A}$ provides a general memory structure.

The automaton is Non-Det, so we can take different transitions in $\mathcal{A}$ when visiting different edges in $\mathcal{G}$ (even if they have the same colour).
If $\mathcal{A}$ is a GFG-Rabin automaton recognizing $\mathcal{F}$, we can do as before:

We build $\mathcal{G} \ltimes \mathcal{A}$, where Eve has a positional strategy. We can transform this strategy in a next-move function

$$\text{next-move}_G : \mathcal{A} \times V_{Eve} \rightarrow E.$$ 

**Conclusion**

$$\text{mem}_{gen}(\mathcal{F}) \leq \text{Size of a minimal GFG Rabin automaton for } \mathcal{F}.$$
[Dziemowski, Jurdziński, Walukiewicz '97] provides a characterization of $\text{mem}_{\text{gen}}(F)$ in terms of a structure called the Zielonka tree of $F$.

Using the Zielonka tree we give a construction of a GFG-Rabin automaton of size $\text{mem}_{\text{gen}}(F)$ recognizing $F$. 

Corollary (C., Colcombet, Lehtinen '21) $\text{mem}_{\text{gen}}(F) =$ Size of a minimal GFG Rabin automaton for $F$. And we can construct one such minimal GFG Rabin automaton "efficiently"!
[Dziembowski, Jurdziński, Walukiewicz ’97] provides a characterization of $\text{mem}_{\text{gen}}(\mathcal{F})$ in terms of a structure called the Zielonka tree of $\mathcal{F}$.

Using the Zielonka tree we give a construction of a GFG-Rabin automaton of size $\text{mem}_{\text{gen}}(\mathcal{F})$ recognizing $\mathcal{F}$.

**Corollary (C., Colcombet, Lehtinen ’21)**

$$\text{mem}_{\text{gen}}(\mathcal{F}) = \text{Size of a minimal GFG Rabin automaton for } \mathcal{F}.$$  

And we can construct one such minimal GFG Rabin automaton “efficiently”!
Good-For-Games Rabin Automata and General Memory

General Memory vs Chromatic Memory

Theorem (C., Colcombet, Lehtinen ’21\(^1\))

There exists a constant \(c > 1\) and a sequence of Muller conditions \(F_1, \ldots, F_n \ldots\) over \(C_1, \ldots, C_n \ldots\), \(|C_n| = n\) such that:

- \(\text{mem}_{\text{gen}}(F_n) = n/2\).
- \(\text{mem}_{\text{chr}}(F_n) = c^n\).

Corollary (C., Colcombet, Lehtinen ’21)

The gap on the size between deterministic and GFG Rabin automata recognizing Muller conditions is exponential.

\(^1\)Thanks to Marthe Bonamy and Pierre Fraigniaud for their help with graph theory!
General Memory vs Chromatic Memory

Theorem (C., Colcombet, Lehtinen ’21\(^1\))

There exists a constant \( c > 1 \) and a sequence of Muller conditions \( \mathcal{F}_1, \ldots, \mathcal{F}_n \ldots \) over \( \mathcal{C}_1, \ldots, \mathcal{C}_n \ldots \), \( |\mathcal{C}_n| = n \) such that:

- \( \text{mem}_{\text{gen}}(\mathcal{F}_n) = \frac{n}{2} \).
- \( \text{mem}_{\text{chr}}(\mathcal{F}_n) = c^n \).

Corollary (C., Colcombet, Lehtinen ’21)

The gap on the size between deterministic and GFG Rabin automata recognizing Muller conditions is exponential.

Thank you!

\(^1\)Thanks to Marthe Bonamy and Pierre Fraigniaud for their help with graph theory!