# Optimal Transformations of Games and Automata using Muller Conditions

Antonio Casares

LaBRI

2 July 2021

Joint work with Thomas Colcombet and Nathanaël Fijalkow.

To appear at ICALP 2021

## Index

## Introduction

We are interested in non-terminating reactive systems:

$$\left.\begin{array}{l}\bullet\ \text{Deterministic } \omega\text{-automata.}\\ \bullet\ \text{Nondeterministic } \omega\text{-automata.}\\ \bullet\ \text{Infinite duration two-players games with}\\ \quad \omega\text{-regular conditions.}\end{array}\right\}\ \text{Transition systems}$$

**Question**

Transition system using a complex acceptance condition $\rightsquigarrow$ Transition system using a simpler acceptance condition

## Introduction

We are interested in non-terminating reactive systems:

- **Deterministic $\omega$-automata.** (This talk)
- Nondeterministic $\omega$-automata.
- Infinite duration two-players games with $\omega$-regular conditions.

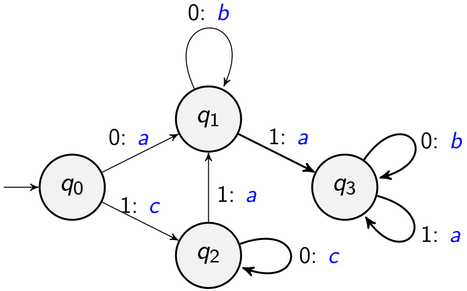
} Transition systems (In the paper)

**Question**

Transition system using a complex acceptance condition $\rightsquigarrow$ Transition system using a simpler acceptance condition

## Deterministic automata over infinite words

- Input alphabet $\Sigma$ $(= \{0, 1\})$.
- Set of states $Q$ (one of them initial).
- Transition function $\delta : Q \times \Sigma \to Q \times \Gamma$.
- Output alphabet $\Gamma$ $(= \{a, b, c\})$.
- **Acceptance condition** $Acc \subseteq \Gamma^\omega$.



$$\text{Input} = 10101000010\cdots \in \Sigma^\omega$$
$$\text{Output} = cababbbbab\cdots \in \Gamma^\omega$$
$$\text{Output} \in Acc\ ?$$

$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\omega$ : the (unique) run produced by reading $w$ is accepted$\}$.

## $\omega$-regular conditions

We are interested in conditions that specify the "asymptotic behaviour" of the system.

Given a word $u \in \Gamma^\omega$ we write:

$$\mathit{Inf}(u) = \{\alpha \in \Gamma \ : \ \alpha \text{ appears infinitely often in } u\}.$$

## Muller conditions

Let $\Gamma$ be a finite alphabet.

A **Muller condition** is given by a family of sets

$$\mathcal{F} \subseteq 2^{\Gamma}.$$

The words accepted by this condition are:

$$Acc_{\mathcal{F}} = \{u \in \Gamma^{\omega} \ : \ Inf(u) \in \mathcal{F}\}.$$

## Parity conditions

Let $\Gamma \subseteq \mathbb{N}$.

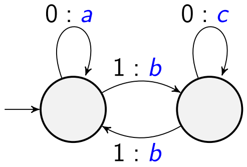
The **parity condition** is:

$$Acc_{Parity} = \{u \in \Gamma^\omega \ : \ \min Inf(u) \text{ is even } \}.$$

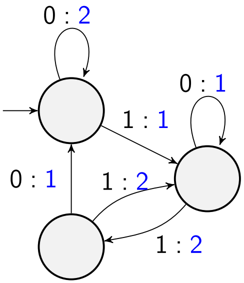
We call the elements of $\Gamma$ **priorities**.

**Remark:** We can reformulate parity conditions as Muller ones.

## Example



Deterministic Muller automaton.
$\mathcal{F}_1 = \{\{a\}, \{b\}\}$.



Deterministic parity automaton.

$$L = \{w \in \{0,1\}^\omega \ : \ \text{eventually } w \text{ only contains } 1,$$
$$\text{or it only contains } 0 \text{ after an even number of occurrences of } 1\}.$$

## Muller vs parity

- Muller conditions are more general than parity ones.
- Parity conditions are easier to deal with:
  - Quasi-polynomial time algorithms for solving parity games.
  - Parity games are positionally determined.
- Every $\omega$-regular language can be recognized by a deterministic parity automaton, and using a parity condition is “the best we can do”. (Strict hierarchy depending on the *number of priorities*).

## Muller vs parity

- Muller conditions are more general than parity ones.
- Parity conditions are easier to deal with:
  - Quasi-polynomial time algorithms for solving parity games.
  - Parity games are positionally determined.
- Every $\omega$-regular language can be recognized by a deterministic parity automaton, and using a parity condition is "the best we can do". (Strict hierarchy depending on the *number of priorities*).

**Objective**

Given a Muller automaton, transform it into a parity automaton

– as small as possible.

– using the least possible number of priorities.

## The classical approach: product by an automaton

**Input:** Automaton $\mathcal{A}$ using a Muller condition $\mathcal{F} \subseteq 2^{\Gamma}$.

## The classical approach: product by an automaton

**Input:** Automaton $\mathcal{A}$ using a **Muller condition** $\mathcal{F} \subseteq 2^{\Gamma}$.

## The classical approach: product by an automaton

**Input:** Automaton $\mathcal{A}$ using a **Muller condition** $\mathcal{F} \subseteq 2^{\Gamma}$.

We can find a deterministic parity automaton $\mathcal{P}$ *recognising the condition*, i.e., for $u \in \Gamma^{\omega}$:

$$u \in \mathcal{L}(\mathcal{P}) \Leftrightarrow \mathit{Inf}(u) \in \mathcal{F}.$$

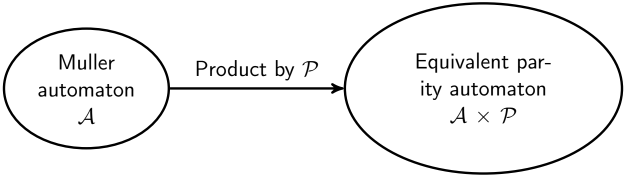
## The classical approach: product by an automaton

**Input:** Automaton $\mathcal{A}$ using a **Muller condition** $\mathcal{F} \subseteq 2^{\Gamma}$.

We can find a deterministic parity automaton $\mathcal{P}$ *recognising the condition*, i.e., for $u \in \Gamma^{\omega}$:

$$u \in \mathcal{L}(\mathcal{P}) \Leftrightarrow \mathit{Inf}(u) \in \mathcal{F}.$$

We can build the product automaton $\mathcal{A} \times \mathcal{P}$, a parity automaton recognizing $\mathcal{L}(\mathcal{A})$ and using the condition from $\mathcal{P}$.


Muller automaton $\mathcal{A}$
Product by $\mathcal{P}$
Equivalent parity automaton $\mathcal{A} \times \mathcal{P}$

***First approach***:

Build a minimal parity automaton recognising a Muller condition!

## The Zielonka tree (Zielonka, '98) and (Dziembowski, Jurdziński, Walukiewicz, '97)

$$\Gamma = \{a, b, c, d\},$$

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

## The Zielonka tree (Zielonka, '98) and (Dziembowski, Jurdziński, Walukiewicz, '97)

$$\Gamma = \{a, b, c, d\},$$

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

Root of the tree $\rightarrow$ $\boxed{\Gamma = \text{a,b,c,d}}$

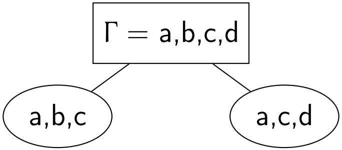
$\boxed{\text{Rejecting set}}$

Accepting set

## The Zielonka tree (Zielonka, ’98) and (Dziembowski, Jurdziński, Walukiewicz, ’97)

$$\Gamma = \{a, b, c, d\},$$

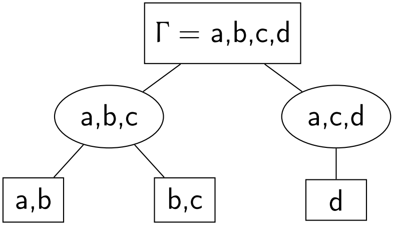
$$\mathcal{F} = \{\{a, b, c\}, \{a, c, d\}, \{a, c\}, \{a, d\}, \{c, d\}, \{a\}, \{c\}\}$$



## The Zielonka tree (Zielonka, ’98) and (Dziembowski, Jurdziński, Walukiewicz, ’97)

$$\Gamma = \{a, b, c, d\},$$

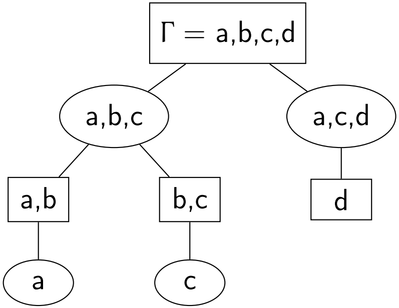
$$\mathcal{F} = \{\{a, b, c\}, \{a, c, d\}, \{a, c\}, \{a, d\}, \{c, d\}, \{a\}, \{c\}\}$$



## The Zielonka tree (Zielonka, '98) and (Dziembowski, Jurdziński, Walukiewicz, '97)

$$\Gamma = \{a, b, c, d\},$$

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



## The Zielonka tree (Zielonka, '98) and (Dziembowski, Jurdziński, Walukiewicz, '97)

$$\Gamma = \{a, b, c, d\},$$

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

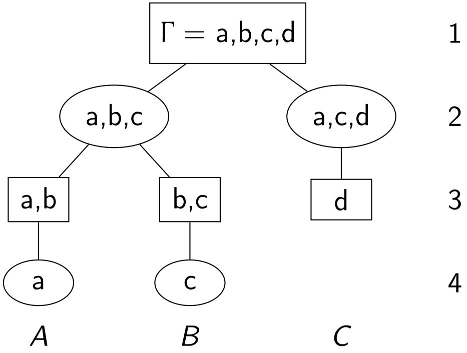
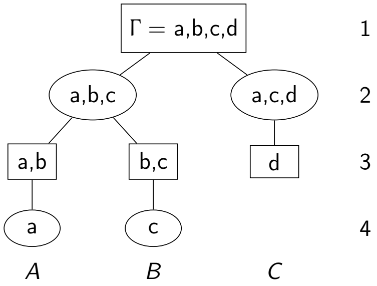
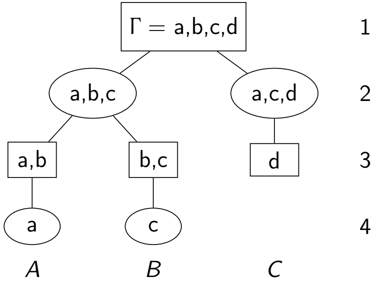


**Figure:** Zielonka Tree $\mathcal{T}_{\mathcal{F}}$

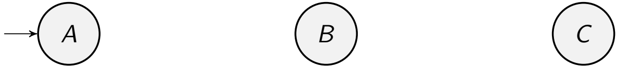
## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$



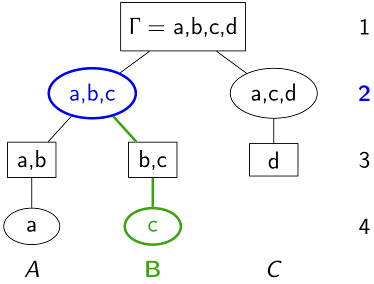
## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$

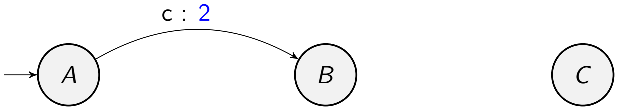


- Set of states: set of branches from the tree.

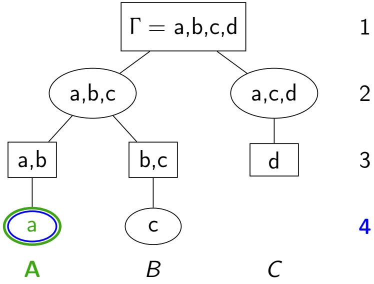


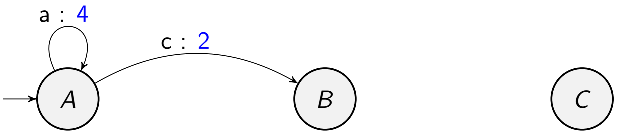
## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$



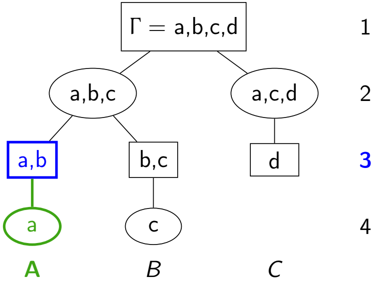


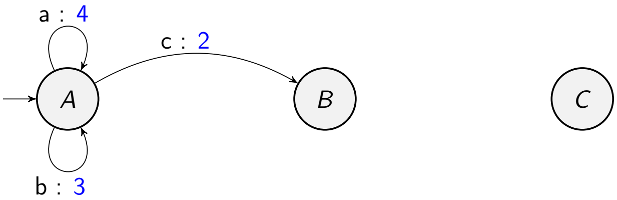
## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$



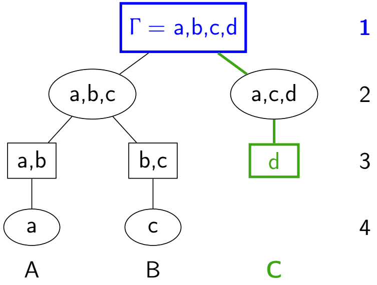


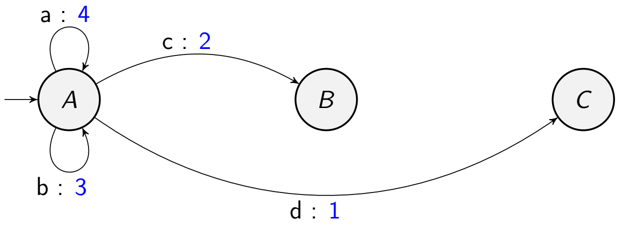
## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$



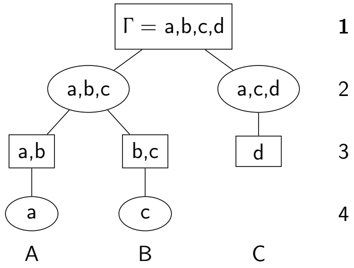


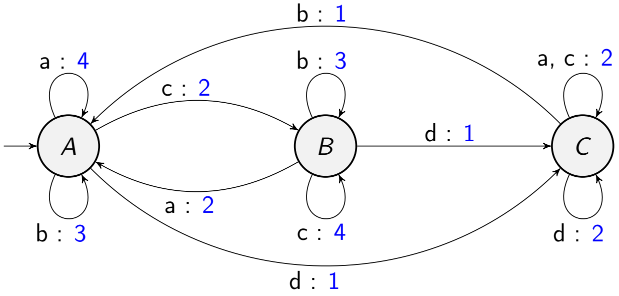
## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$





## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$





## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$: Correctness

We write $\mathcal{Z}_{\mathcal{F}}$ to refer to the Zielonka tree automaton of the condition $\mathcal{F}$.

**Proposition**

$$\mathcal{L}(\mathcal{Z}_{\mathcal{F}}) = \{u \in \Gamma^\omega \ :\ \mathit{Inf}(u) \in \mathcal{F}\}.$$

## The Zielonka tree automaton $\mathcal{Z}_{\mathcal{F}}$: Optimality

**Theorem (Independently proven by Meyer and Sickert)**

*Let $\mathcal{P}$ be a deterministic parity automaton recognizing the Muller condition $\mathcal{F}$. Then,*

1. $|\mathcal{Z}_{\mathcal{F}}| \leq |\mathcal{P}|$.
2. *$\mathcal{P}$ uses as least as many priorities as $\mathcal{Z}_{\mathcal{F}}$.*

**Proof idea:**

1. We recursively decompose $\mathcal{P}$ following the Zielonka tree.
2. The length of a branch determines the number of alternations that we need to take into account.

## Minimisation of automata

**Corollary**

*We can minimise parity automata recognising Muller conditions in polynomial time.*

In general, minimisation is:

- NP-complete for state-based parity automata.
- Unknown for transition-based parity automata.

## Conclusion

Given a Muller automaton $\mathcal{A}$ using condition $\mathcal{F}$, we have the transformation

$$\mathcal{A} \times \mathcal{Z}_{\mathcal{F}}$$

that gives a parity automaton recognizing the same language.

## Conclusion

Given a Muller automaton $\mathcal{A}$ using condition $\mathcal{F}$, we have the transformation

$$\mathcal{A} \times \mathcal{Z}_{\mathcal{F}}$$

that gives a parity automaton recognizing the same language.

But... Can we do better?

## Conclusion

Given a Muller automaton $\mathcal{A}$ using condition $\mathcal{F}$, we have the transformation

$$\mathcal{A} \times \mathcal{Z}_{\mathcal{F}}$$

that gives a parity automaton recognizing the same language.

But... Can we do better?

**YES!**

## Why isn't the product approach optimal?

➔ **The product approach does not take into account the structure of the input automaton $\mathcal{A}$.**

Intuitively, this is due to:

- $\mathcal{A}$ might not use the full power of the considered Muller condition.
- Different parts of $\mathcal{A}$ make use of the condition in different ways.

## Why isn't the product approach optimal?

➔ **The product approach does not take into account the structure of the input automaton $\mathcal{A}$.**

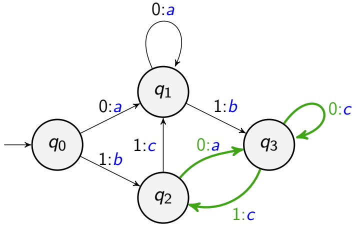
Intuitively, this is due to:

- $\mathcal{A}$ might not use the full power of the considered Muller condition.
- Different parts of $\mathcal{A}$ make use of the condition in different ways.

We propose a generalisation analysing the structure of the given Muller automaton.

## Loops of an automaton

A *loop* of an automaton $\mathcal{A}$ is a set of edges that forms a closed path (not necessarily simple).



Given a Muller condition over $\mathcal{A}$ we have a mapping

$$f : \mathcal{L}oops(\mathcal{A}) \to \{Accept, Reject\}.$$

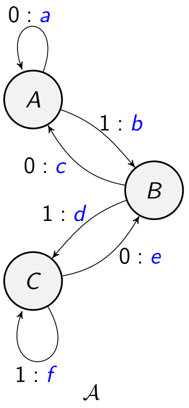
## The Alternating Cycle Decomposition

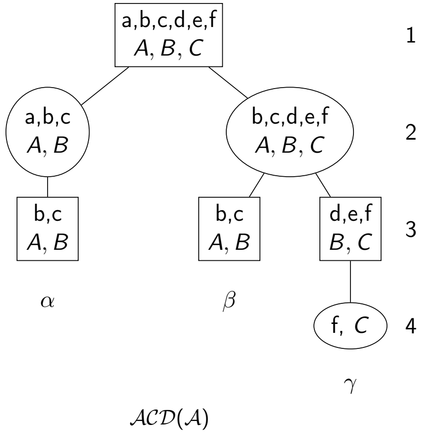
The Alternating Cycle Decomposition

We recursively build a tree (as the Zielonka tree), but considering the **loops of the automaton** instead of the colours of the condition.

## The Alternating Cycle Decomposition

In blue the names of the edges of the automaton.





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

## The Alternating Cycle Decomposition Transformation

We can use this structure to define a parity automaton equivalent to $\mathcal{A}$.

We denote this parity automaton $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$.

**Claim**

*This transformation is optimal.*

But... What is an **optimal transformation**?

## Morphisms of deterministic automata

**Definition (Morphism of automata)**

Let $\mathcal{A}$ and $\mathcal{B}$ be two deterministic automata over an input alphabet $\Sigma$. A *morphism* is a mapping of states

$$\varphi\colon \mathcal{A} \to \mathcal{B}$$

that verifies two local conditions:

1. Preserves the initial state.
2. Preserves transitions $(\delta_A(q,x) = q' \;\Rightarrow\; \delta_B(\varphi(q),x) = \varphi(q'))$.

and the following global condition:

3. A run $\varrho$ in $\mathcal{A}$ is accepting if and only if $\varphi(\varrho)$ is accepting.

## Morphisms of deterministic automata

### Definition (Morphism of automata)

Let $\mathcal{A}$ and $\mathcal{B}$ be two deterministic automata over an input alphabet $\Sigma$. A *morphism* is a mapping of states

$$\varphi\colon \mathcal{A} \to \mathcal{B}$$

that verifies two local conditions:

1. Preserves the initial state.
2. Preserves transitions ($\delta_A(q, x) = q' \Rightarrow \delta_B(\varphi(q), x) = \varphi(q')$).

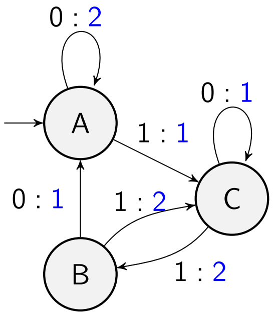
and the following global condition:

3. A run $\varrho$ in $\mathcal{A}$ is accepting if and only if $\varphi(\varrho)$ is accepting.

### Proposition

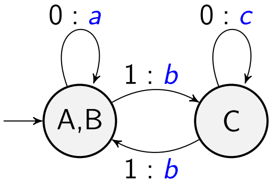
*If there exists a morphism $\varphi\colon \mathcal{A} \to \mathcal{B}$, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.*

## Example of morphism



Parity automaton.

$$\xrightarrow{\varphi}$$



$\mathcal{F}_1 = \{\{a\}, \{b\}\}$.

## Optimality of the ACD-transformation

Let $\mathcal{A}$ be a Muller automaton and let $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$ be the parity automaton obtained using the ACD.

**Proposition (Correctness)**

*There is a morphism* $\varphi\colon \mathcal{P}_{\mathcal{ACD}(\mathcal{A})} \to \mathcal{A}$.

## Optimality of the ACD-transformation

Let $\mathcal{A}$ be a Muller automaton and let $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$ be the parity automaton obtained using the ACD.

**Proposition (Correctness)**

*There is a morphism* $\varphi\colon \mathcal{P}_{\mathcal{ACD}(\mathcal{A})} \to \mathcal{A}$.

**Theorem (Optimality)**

*Let* $\mathcal{P}'$ *be a parity automaton such that there is a morphism of automata* $\varphi\colon \mathcal{P}' \to \mathcal{A}$. *Then:*

1. $|\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}| \leq |\mathcal{P}'|$.
2. $\mathcal{P}'$ *uses at least as many priorities as* $\mathcal{P}_{\mathcal{ACD}(\mathcal{A})}$.

## Nondeterministic automata and games

**Remark (Locally bijective morphisms as witnesses of transformations)**

*In order to state the correctness and optimality for nondeterministic automata and games we have to introduce the notion of* ***locally bijective morphism*** *(similar to bisimulation).*

➔*Locally bijective morphisms preserve all the desired semantic properties.*

## Application: Relabelling automata with simpler conditions

Let $\mathcal{A}$ be a Muller automaton. We say that we can *put a parity condition on top of* $\mathcal{A}$, if we can relabel the transitions of $\mathcal{A}$ with priorities in $\mathbb{N}$, obtaining an automaton $\mathcal{A}'$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}').$$

## Application: Relabelling automata with simpler conditions

Let $\mathcal{A}$ be a Muller automaton. We say that we can *put a parity condition on top of* $\mathcal{A}$, if we can relabel the transitions of $\mathcal{A}$ with priorities in $\mathbb{N}$, obtaining an automaton $\mathcal{A}'$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}').$$

***When can we put a parity condition on top of a Muller automaton?***

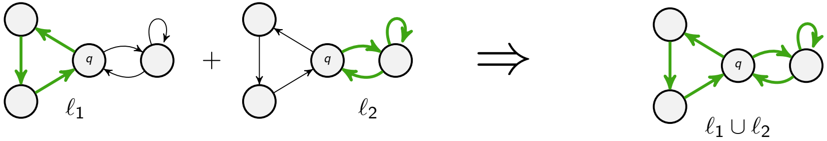
## Application: Relabelling automata with simpler conditions

The ACD allows us to prove the following result:

**Theorem (Relabelling with parity conditions)**

*Let $\mathcal{A}$ be a Muller automaton. The following are equivalent:*

- *We can define a parity condition on top of $\mathcal{A}$ obtaining an equivalent automaton.*
- *For any pair of loops $\ell_1$, $\ell_2$ with a common state, if both are accepting or rejecting, so is their union.*



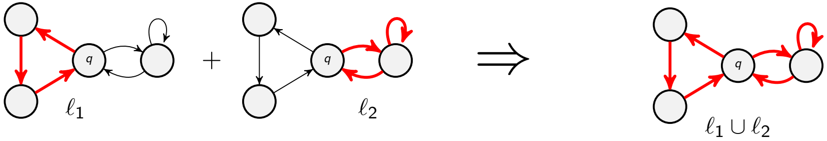
## Application: Relabelling automata with simpler conditions

The ACD allows us to prove the following result:

**Theorem (Relabelling with parity conditions)**

*Let $\mathcal{A}$ be a Muller automaton. The following are equivalent:*

- *We can define a parity condition on top of $\mathcal{A}$ obtaining an equivalent automaton.*
- *For any pair of loops $\ell_1$, $\ell_2$ with a common state, if both are accepting or rejecting, so is their union.*



## Application: Relabelling automata with simpler conditions

The ACD allows us to prove the following result:

**Theorem (Rellabeling with Rabin conditions)**

*Let $\mathcal{A}$ be a Muller automaton. The following are equivalent:*

- *We can define a Rabin condition on top of $\mathcal{A}$ obtaining an equivalent automaton.*
- *For any pair of loops $\ell_1$, $\ell_2$ with a common state, if both are rejecting, so is their union.*

## Application: Relabelling automata with simpler conditions

**Corollary (Boker,Kupferman,Steinitz '10)**

*We can put a parity condition on top of a deterministic Muller automaton $\mathcal{A}$ if and only if we can put both Rabin and Streett conditions on top of it.*

**Corollary (C. '21)**

*Arena-independent memories for games using a given Muller condition coincide with Rabin automata for this condition.*

**Corollary (C. '21)**

*Minimisation of transition-based Rabin automata is* $\mathtt{NP}$*-complete.*

## Application: Determinisation of Büchi automata

**Theorem (Piterman 06')**

*Given a non-deterministic Büchi automaton $\mathcal{B}$, there is a construction that produces a deterministic automaton $\mathcal{P}_\mathcal{B}$ such that*

$$\mathcal{L}(\mathcal{P}_\mathcal{B}) = \mathcal{L}(\mathcal{B}).$$

We can see this construction in two steps (Schewe 09'):

$$\mathcal{B} \quad \longrightarrow \quad \mathcal{M}_\mathcal{B} \quad \longrightarrow \quad \mathcal{P}_\mathcal{B}$$

where $\mathcal{M}_\mathcal{B}$ is a Muller automaton.

We can substitute the second step by the ACD-transformation, obtaining a smaller parity automaton using less priorities.

- There are examples (most of them) where the size is strictly smaller.
- For the worst case we obtain the same construction.

## Conclusions

➔ The ACD provides an optimal transformation of Muller automata into parity automata.

➔ The ACD captures the structure of automata and games and the interplay with the acceptance condition. This can be used to prove theoretical results about these transition systems. (See latest results about the chromatic memory requirements for Muller games on Arxiv).

**Thank you!**