

Hierarchies of Abstract Machines

Jérémie Cabessa

University Joseph Fourier – Grenoble 1

13 February 2009

Outline

1 Introduction

2 Muller automata

3 The Wagner hierarchy

4 More complex hierarchies

5 Conclusion

Outline

1 Introduction

2 Muller automata

3 The Wagner hierarchy

4 More complex hierarchies

5 Conclusion

Outline

- 1 Introduction**
- 2 Muller automata**
- 3 The Wagner hierarchy**
- 4 More complex hierarchies
- 5 Conclusion

Outline

- 1** Introduction
- 2** Muller automata
- 3** The Wagner hierarchy
- 4** More complex hierarchies
- 5 Conclusion

Outline

- 1 Introduction
- 2 Muller automata
- 3 The Wagner hierarchy
- 4 More complex hierarchies
- 5 Conclusion



The fields of artificial neural networks and theoretical computer science are very interconnected.

Muller automata

- Muller automata are abstract models of very rudimentary computers.
- Every other more sophisticated model is an extension of Muller automata.

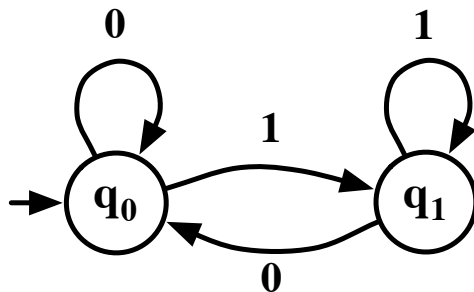
Muller automata

- Muller automata are abstract models of very rudimentary computers.
- Every other more sophisticated model is an extension of Muller automata.

Muller automata

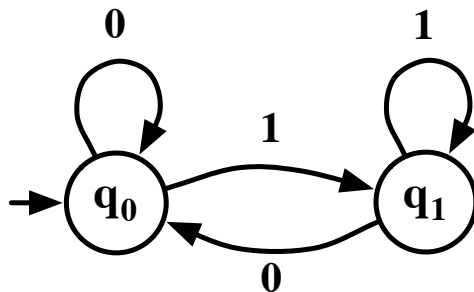
- Muller automata are abstract models of very rudimentary computers.
- Every other more sophisticated model is an extension of Muller automata.

Muller automaton (deterministic)



$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Muller automaton (deterministic)



$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

input 001111111111.....

accepted

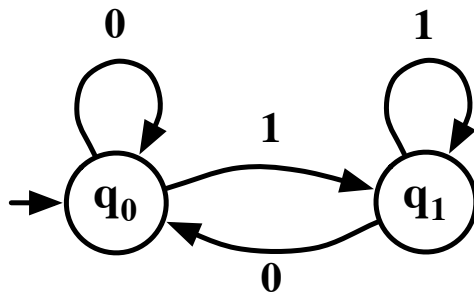
input 110000000000.....

rejected

input 010101010101.....

accepted

Muller automaton (deterministic)



$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

input 001111111111.....

accepted

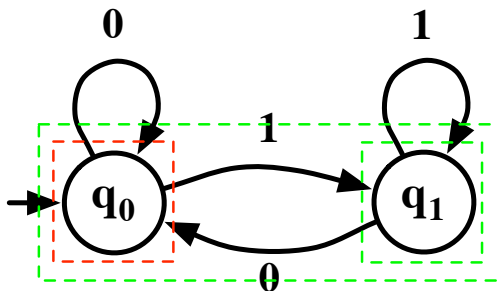
input 110000000000.....

rejected

input 010101010101.....

accepted

Muller automaton (deterministic)



$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

input 001111111111.....

accepted

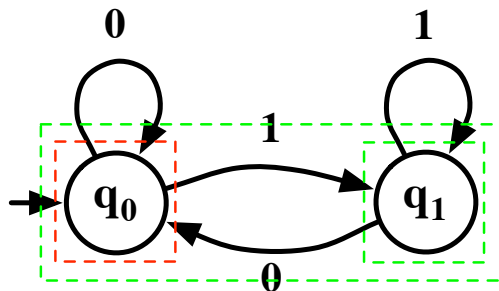
input 110000000000.....

rejected

input 010101010101.....

accepted

Muller automaton (deterministic)



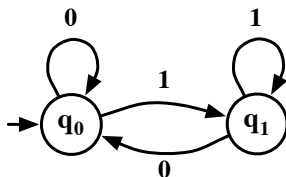
$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

The accepted words are exactly those containing infinitely many 1's. This is the *language recognized by this automaton*.

Definition

A finite *Muller automaton* is a tuple $\mathcal{A} = (Q, A, \delta, i, \mathcal{T})$, where

- Q is a finite set of states,
- A is an alphabet,
- $\delta : Q \times A \longrightarrow Q$ is the transition function,
- $i \in Q$ is the initial state,
- $\mathcal{T} \subseteq \mathcal{P}(Q)$ is the table of \mathcal{A} .

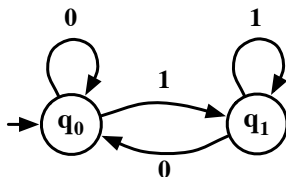


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

A finite *Muller automaton* is a tuple $\mathcal{A} = (Q, A, \delta, i, \mathcal{T})$, where

- Q is a finite set of states,
- A is an alphabet,
- $\delta : Q \times A \longrightarrow Q$ is the transition function,
- $i \in Q$ is the initial state,
- $\mathcal{T} \subseteq \mathcal{P}(Q)$ is the table of \mathcal{A} .

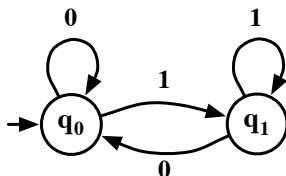


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

A finite *Muller automaton* is a tuple $\mathcal{A} = (Q, A, \delta, i, \mathcal{T})$, where

- Q is a finite set of states,
- A is an alphabet,
- $\delta : Q \times A \longrightarrow Q$ is the transition function,
- $i \in Q$ is the initial state,
- $\mathcal{T} \subseteq \mathcal{P}(Q)$ is the table of \mathcal{A} .

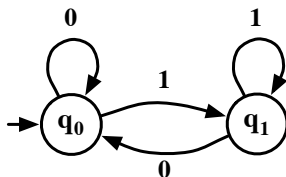


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

A finite *Muller automaton* is a tuple $\mathcal{A} = (Q, A, \delta, i, \mathcal{T})$, where

- Q is a finite set of states,
- A is an alphabet,
- $\delta : Q \times A \longrightarrow Q$ is the transition function,
- $i \in Q$ is the initial state,
- $\mathcal{T} \subseteq \mathcal{P}(Q)$ is the table of \mathcal{A} .

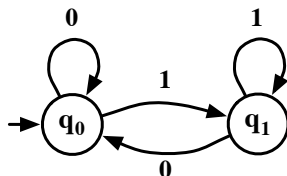


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

A finite *Muller automaton* is a tuple $\mathcal{A} = (Q, A, \delta, i, \mathcal{T})$, where

- Q is a finite set of states,
- A is an alphabet,
- $\delta : Q \times A \longrightarrow Q$ is the transition function,
- $i \in Q$ is the initial state,
- $\mathcal{T} \subseteq \mathcal{P}(Q)$ is the table of \mathcal{A} .

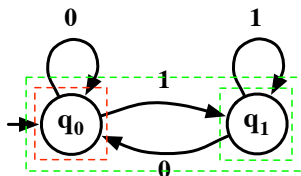


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

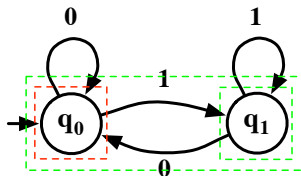
A finite *Muller automaton* is a tuple $\mathcal{A} = (Q, A, \delta, i, \mathcal{T})$, where

- Q is a finite set of states,
- A is an alphabet,
- $\delta : Q \times A \longrightarrow Q$ is the transition function,
- $i \in Q$ is the initial state,
- $\mathcal{T} \subseteq \mathcal{P}(Q)$ is the table of \mathcal{A} .



$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

An infinite input w induces an infinite path in $\mathcal{A} \dots$ An input w is accepted by \mathcal{A} iff $States_\infty(w) \in \mathcal{T}$.

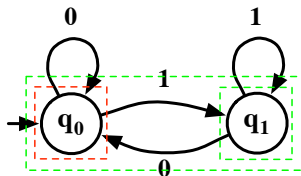


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

The set $L(\mathcal{A})$ of all infinite words accepted by \mathcal{A} is called the *language recognized by \mathcal{A}* .

An infinite input w induces an infinite path in $\mathcal{A} \dots$ An input w is *accepted* by \mathcal{A} iff $States_\infty(w) \in \mathcal{T}$.

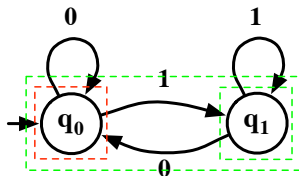


$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

The set $L(\mathcal{A})$ of all infinite words accepted by \mathcal{A} is called the *language recognized by \mathcal{A}* .

An infinite input w induces an infinite path in $\mathcal{A} \dots$ An input w is *accepted* by \mathcal{A} iff $States_{\infty}(w) \in \mathcal{T}$.



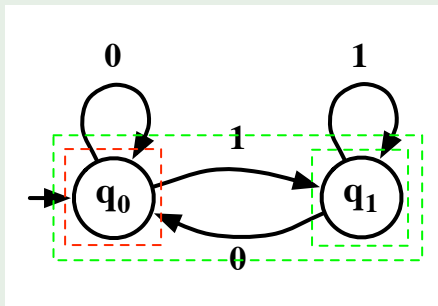
$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

Definition

The set $L(\mathcal{A})$ of all infinite words accepted by \mathcal{A} is called the *language recognized by \mathcal{A}* .

Example

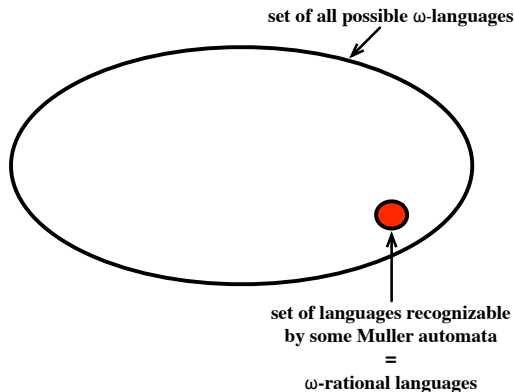
Consider the following automaton \mathcal{A} . Then $L(\mathcal{A})$ is the set of words containing infinitely many 1's.



$$\text{table } \mathcal{T} = \{\{q_0, q_1\}, \{q_1\}\}$$

From now on, we will generally identify a Muller automaton \mathcal{A} with its corresponding ω -language $L(\mathcal{A})$.

From now on, we will generally identify a Muller automaton \mathcal{A} with its corresponding ω -language $L(\mathcal{A})$.



The Wagner hierarchy

- A very refined topological classification of ω -rational languages;
- or equivalently, a very refined classification of Muller automata.

The Wagner hierarchy

- A very refined topological classification of ω -rational languages;
- or equivalently, a very refined classification of Muller automata.

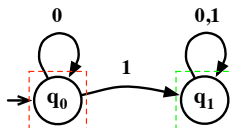
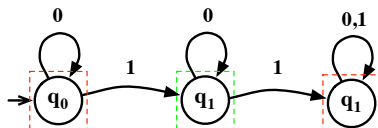
The Wagner hierarchy

- A very refined topological classification of ω -rational languages;
- or equivalently, a very refined classification of Muller automata.

Every classification comes from a partial order (\leq) between the objects that we consider. We will define a specific partial order \leq_W between Muller automata.

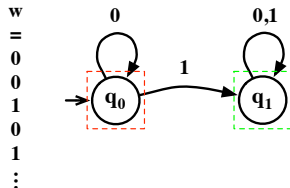
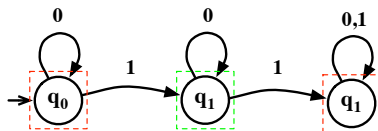
Every classification comes from a partial order (\leq) between the objects that we consider. We will define a specific partial order \leq_w between Muller automata.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

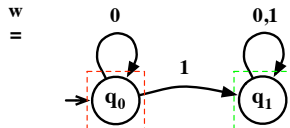
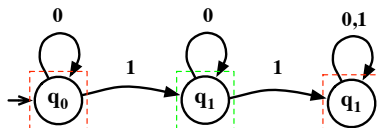
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

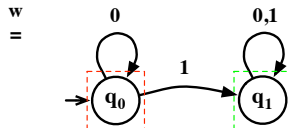
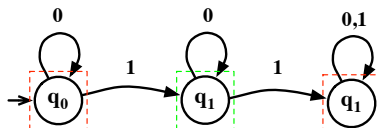
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

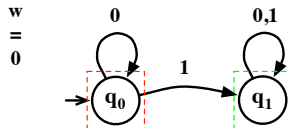
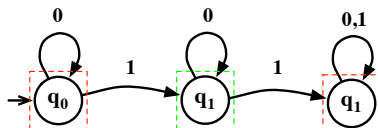
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

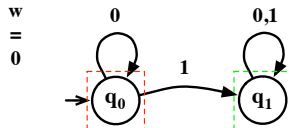
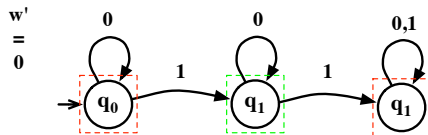
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

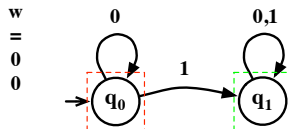
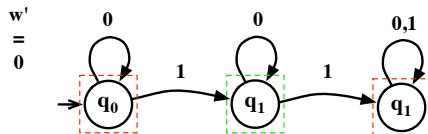
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

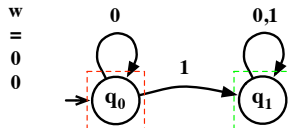
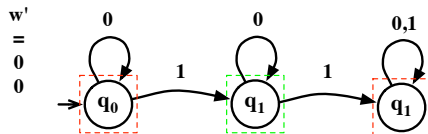
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

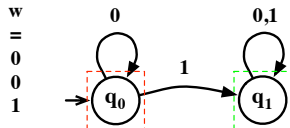
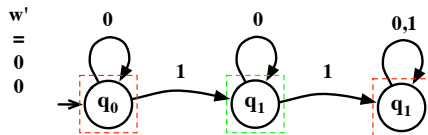
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

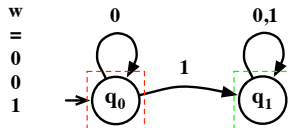
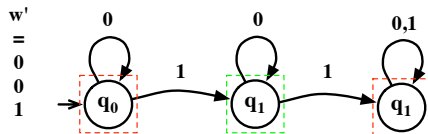
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

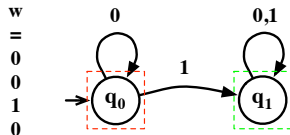
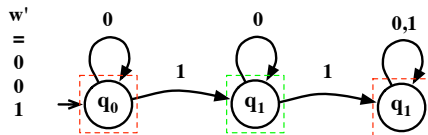
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

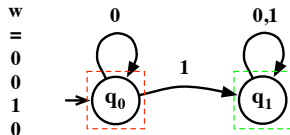
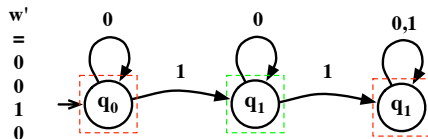
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

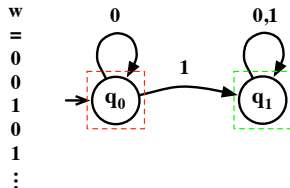
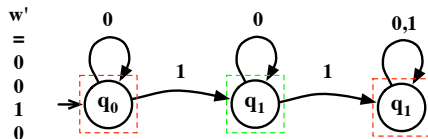
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

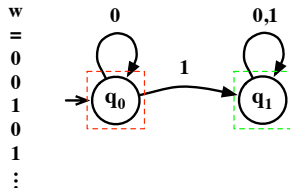
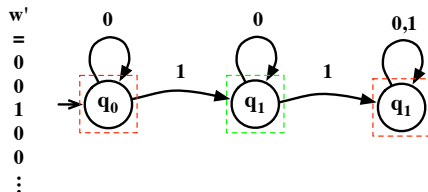
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

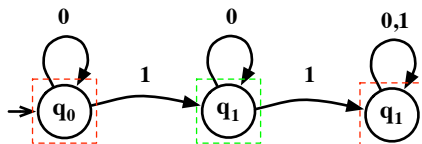
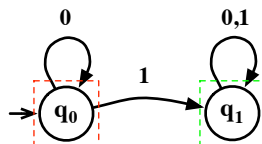
Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

For every infinite input disclosed to \mathcal{A} letter by letter, it is possible to simultaneously construct letter by letter an infinite input for \mathcal{B} of the same acceptance.

Automaton \mathcal{A} Automaton \mathcal{B} 

Automaton \mathcal{B} is able to “mimic” \mathcal{A} correctly. Any behaviour of \mathcal{A} can be simulated by \mathcal{B} . We say that \mathcal{A} reduces to \mathcal{B} , denoted by $\mathcal{A} \leq_w \mathcal{B}$.

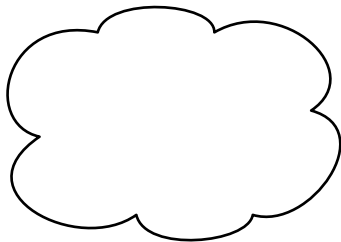
However $\mathcal{B} \not\leq_W \mathcal{A}$, that is \mathcal{A} is not able to mimic \mathcal{B} correctly
(consider the \mathcal{A} -input 1100000...)

Automaton \mathcal{B} Automaton \mathcal{A} 

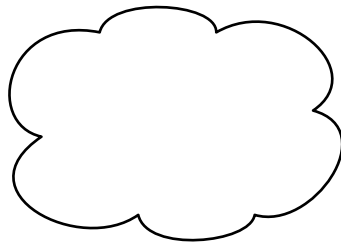
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



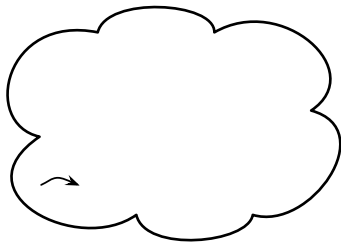
Automaton \mathcal{B}



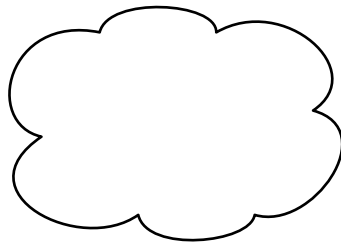
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



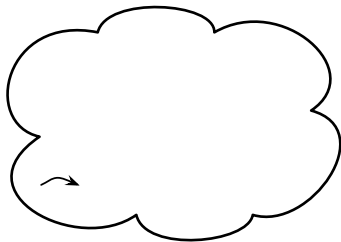
Automaton \mathcal{B}



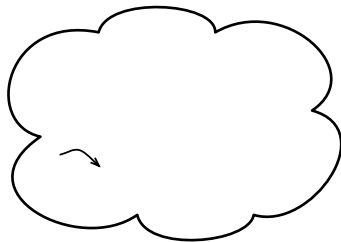
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



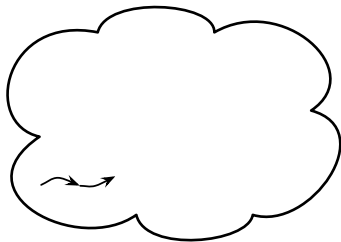
Automaton \mathcal{B}



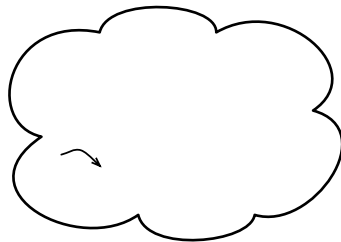
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



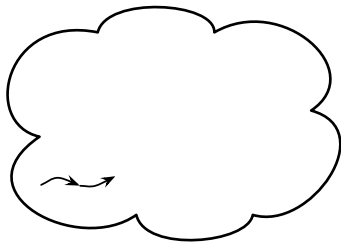
Automaton \mathcal{B}



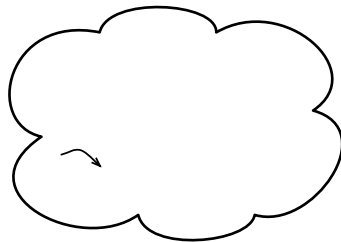
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



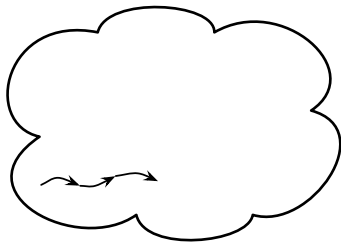
Automaton \mathcal{B}



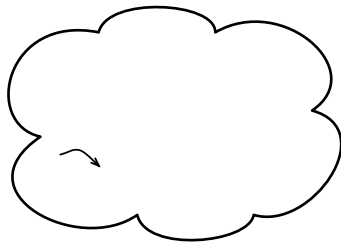
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



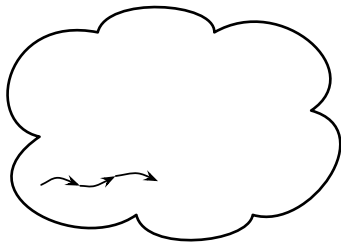
Automaton \mathcal{B}



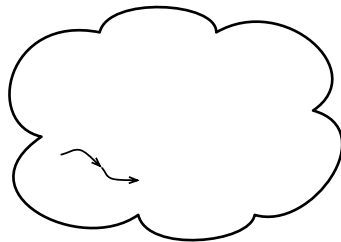
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



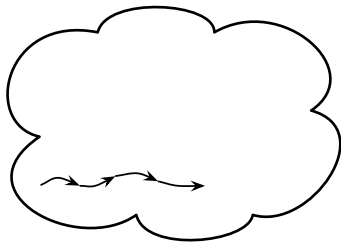
Automaton \mathcal{B}



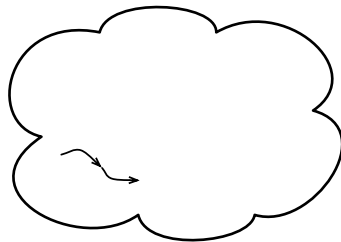
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



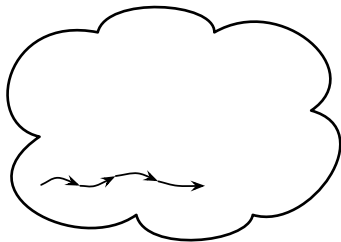
Automaton \mathcal{B}



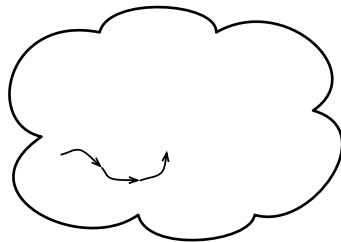
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



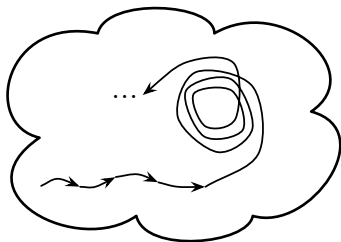
Automaton \mathcal{B}



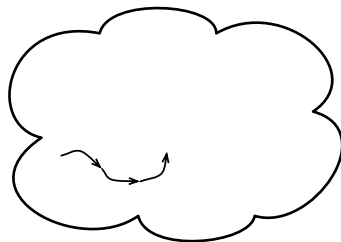
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



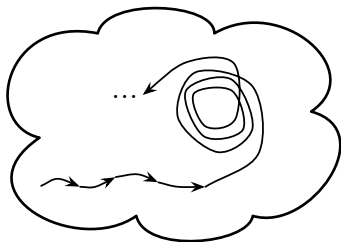
Automaton \mathcal{B}



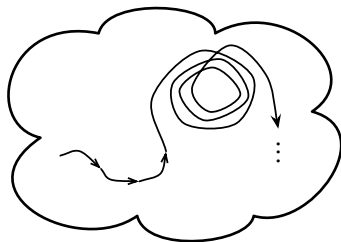
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



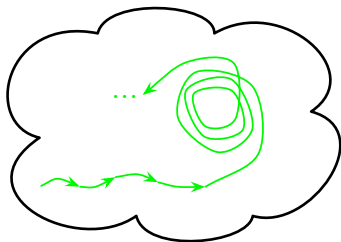
Automaton \mathcal{B}



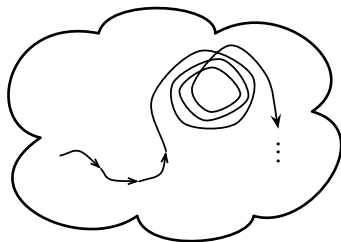
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



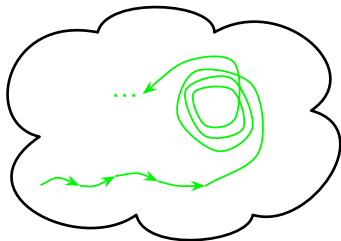
Automaton \mathcal{B}



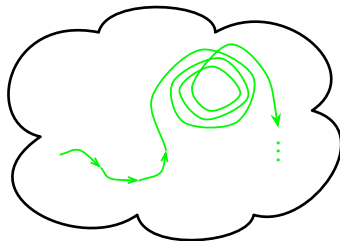
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



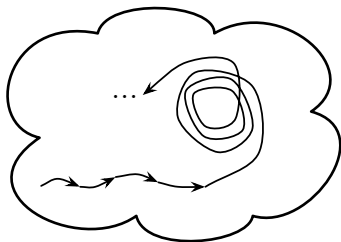
Automaton \mathcal{B}



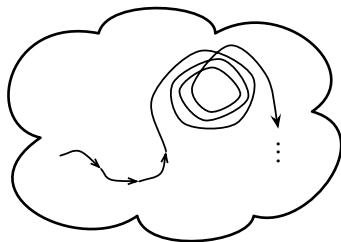
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



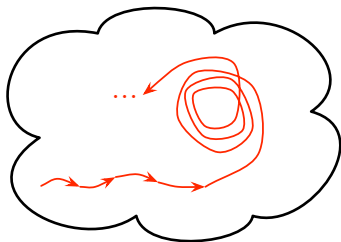
Automaton \mathcal{B}



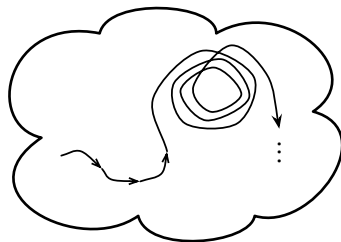
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



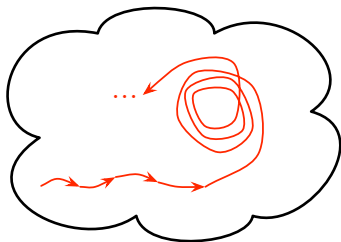
Automaton \mathcal{B}



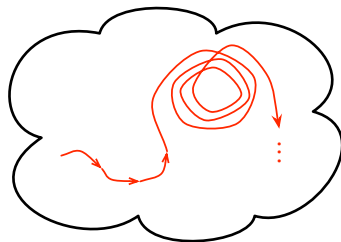
Definition

Let \mathcal{A} and \mathcal{B} be two Muller automata. We say that $\mathcal{A} \leq_w \mathcal{B}$ iff for every \mathcal{A} -input disclosed letter by letter, it is possible to simultaneously construct letter by letter (skipping moves are allowed) a \mathcal{B} -input of the same acceptance.

Automaton \mathcal{A}



Automaton \mathcal{B}



Definition (alt. def. of the same reduction relation)

We say that $\mathcal{A} \leq_w \mathcal{B}$ iff there exists a continuous function f such that $L(\mathcal{A}) = f^{-1}(L(\mathcal{B}))$.

- We set $\mathcal{A} <_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.
- We set $\mathcal{A} \equiv_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \leq_w \mathcal{A}$.
- We set $\mathcal{A} \not\equiv_w \mathcal{B}$ if both $\mathcal{A} \not\leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.

Definition (alt. def. of the same reduction relation)

We say that $\mathcal{A} \leq_w \mathcal{B}$ iff there exists a continuous function f such that $L(\mathcal{A}) = f^{-1}(L(\mathcal{B}))$.

- We set $\mathcal{A} <_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.
- We set $\mathcal{A} \equiv_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \leq_w \mathcal{A}$.
- We set $\mathcal{A} \not\equiv_w \mathcal{B}$ if both $\mathcal{A} \not\leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.

Definition (alt. def. of the same reduction relation)

We say that $\mathcal{A} \leq_w \mathcal{B}$ iff there exists a continuous function f such that $L(\mathcal{A}) = f^{-1}(L(\mathcal{B}))$.

- We set $\mathcal{A} <_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.
- We set $\mathcal{A} \equiv_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \leq_w \mathcal{A}$.
- We set $\mathcal{A} \not\equiv_w \mathcal{B}$ if both $\mathcal{A} \not\leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.

Definition (alt. def. of the same reduction relation)

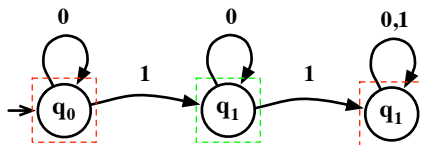
We say that $\mathcal{A} \leq_w \mathcal{B}$ iff there exists a continuous function f such that $L(\mathcal{A}) = f^{-1}(L(\mathcal{B}))$.

- We set $\mathcal{A} <_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.
- We set $\mathcal{A} \equiv_w \mathcal{B}$ if both $\mathcal{A} \leq_w \mathcal{B}$ and $\mathcal{B} \leq_w \mathcal{A}$.
- We set $\mathcal{A} \not\equiv_w \mathcal{B}$ if both $\mathcal{A} \not\leq_w \mathcal{B}$ and $\mathcal{B} \not\leq_w \mathcal{A}$.

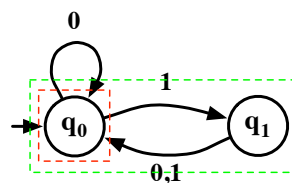
Example (strict reduction)

Here, $\mathcal{A} \leq_W \mathcal{B}$ but $\mathcal{B} \not\leq_W \mathcal{A}$, therefore $\mathcal{A} <_W \mathcal{B}$.

Automaton \mathcal{A}



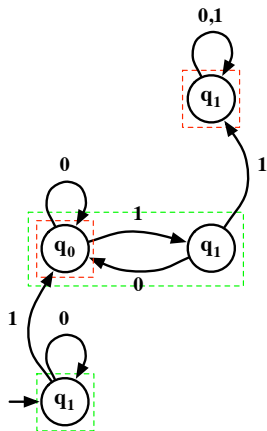
Automaton \mathcal{B}



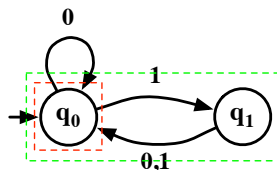
Example (equivalence)

Here, $\mathcal{A} \leq_W \mathcal{B}$ and $\mathcal{B} \leq_W \mathcal{A}$, therefore $\mathcal{A} \equiv_W \mathcal{B}$.

Automaton \mathcal{A}



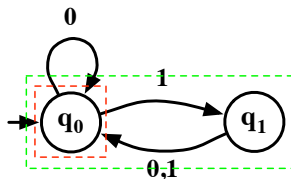
Automaton \mathcal{B}



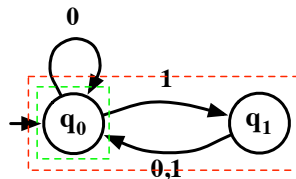
Example (incomparability)

Here, $\mathcal{A} \not\leq_W \mathcal{B}$ and $\mathcal{B} \not\leq_W \mathcal{A}$, therefore $\mathcal{A} \not\equiv_W \mathcal{B}$.

Automaton \mathcal{A}



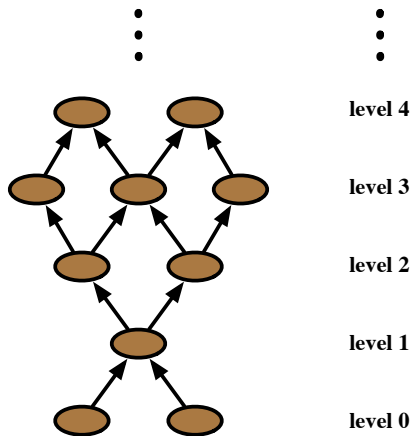
Automaton \mathcal{B}



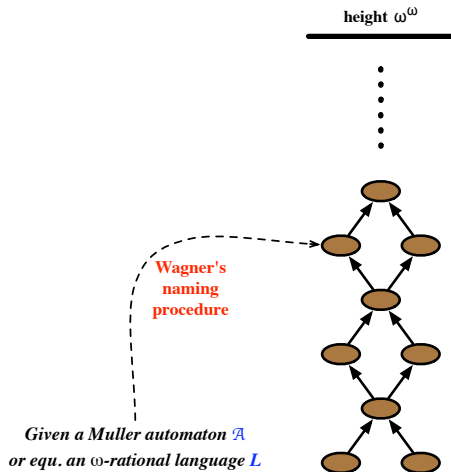
Definition

The collection of all Muller automata (or ω -rational languages) ordered by the relation \leq_w is called the *Wagner hierarchy*.

We want to describe the shape and compute the height of the Wagner hierarchy . . . (circle are the \equiv_W -equivalence classes and arrows are the $\text{strict}_{<_W}$ -reduction)

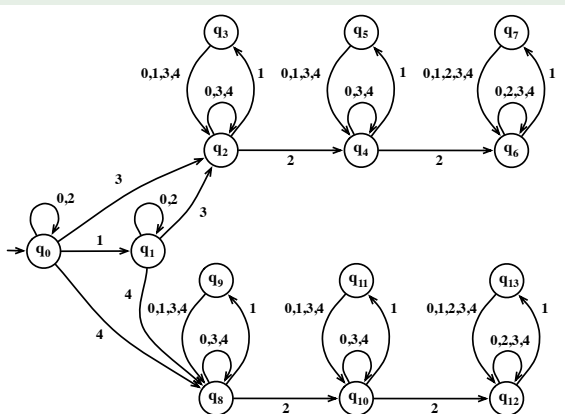


The Wagner hierarchy is decidable.



Example

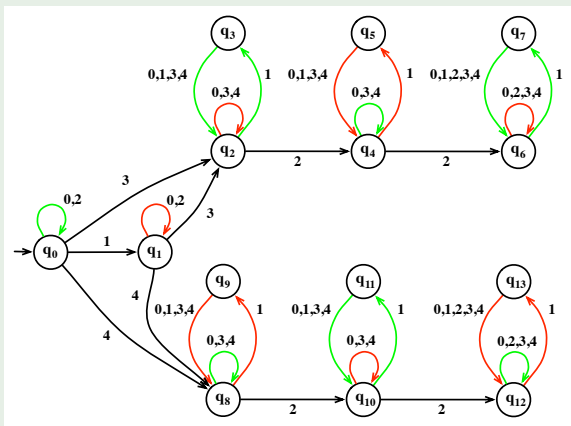
The following Muller automata has height $\omega \cdot 3 + 2$ in the Wagner hierarchy.



$$T = \{\{q_0\}, \{q_2, q_3\}, \{q_4\}, \{q_6, q_7\}, \{q_8\}, \{q_{10}, q_{11}\}, \{q_{12}\}\}$$

Example

The following Muller automata has height $\omega \cdot 3 + 2$ in the Wagner hierarchy.



$$\mathcal{T} = \{\{q_0\}, \{q_2, q_3\}, \{q_4\}, \{q_6, q_7\}, \{q_8\}, \{q_{10}, q_{11}\}, \{q_{12}\}\}$$

Main kind abstract Machines reading infinite words:

- Deterministic Muller automata
- Deterministic Muller Counter automata
- Deterministic Muller Pushdown automata
- Deterministic Muller Turing machines

Main kind abstract Machines reading infinite words:

- Deterministic Muller automata
- Deterministic Muller Counter automata
- Deterministic Muller Pushdown automata
- Deterministic Muller Turing machines

Main kind abstract Machines reading infinite words:

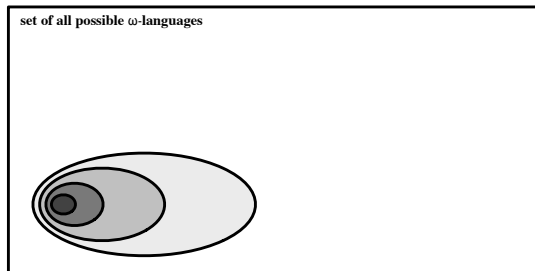
- Deterministic Muller automata
- Deterministic Muller Counter automata
- Deterministic Muller Pushdown automata
- Deterministic Muller Turing machines

Main kind abstract Machines reading infinite words:

- Deterministic Muller automata
- Deterministic Muller Counter automata
- Deterministic Muller Pushdown automata
- Deterministic Muller Turing machines

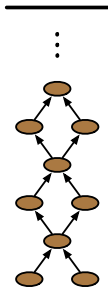
Main kind abstract Machines reading infinite words:

- Deterministic Muller automata
- Deterministic Muller Counter automata
- Deterministic Muller Pushdown automata
- Deterministic Muller Turing machines

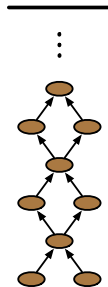


- ω -languages recognized by Muller automata
- ω -languages recognized by Muller counter automata
- ω -languages recognized by Muller pushdown automata
- ω -languages recognized by Muller Turing machines

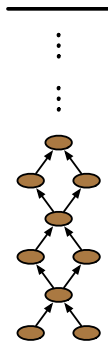
DetMA

height ω^ω Decidability pb:
solved

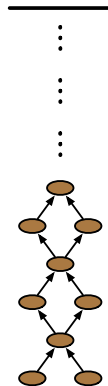
DetMCA

height $(\omega^2)^\omega = \omega^\omega$ Decidability pb:
solved

DetMPDA

height $(\omega^\omega)^\omega$ Decidability pb:
unsolved

DetMTM

height $(\omega_1^{CK})^\omega$ Decidability pb:
undecidable

- The reduction relation \leq_w leads to very refined transfinite hierarchies of abstract machines.
- These hierarchies of abstract machines represent a classification of all possible algorithms.
- These hierarchies of abstract machines are just specific restrictions of the whole topological hierarchy of all ω -languages.

- The reduction relation \leq_w leads to very refined transfinite hierarchies of abstract machines.
- These hierarchies of abstract machines represent a classification of all possible algorithms.
- These hierarchies of abstract machines are just specific restrictions of the whole topological hierarchy of all ω -languages.

- The reduction relation \leq_w leads to very refined transfinite hierarchies of abstract machines.
- These hierarchies of abstract machines represent a classification of all possible algorithms.
- These hierarchies of abstract machines are just specific restrictions of the whole topological hierarchy of all ω -languages.