The bridge between regular cost functions and omega-regular languages

Thomas Colcombet and Nathanaël Fijalkow
ICALP
July 15 2016
The bridge between regular cost functions and omega-regular languages

Thomas Colcombet and Nathanaël Fijalkow
ICALP
July 15 2016
The bridge between regular cost functions and omega-regular languages

Thomas Colcombet and Nathanaël Fijalkow
ICALP
July 15 2016
The bridge between regular cost functions and omega-regular languages

A way to transfer results and constructions

Regular languages over words of length ω

A quantitative extension of regular languages

Thomas Colcombet and Nathanaël Fijalkow
ICALP
July 15 2016
Regular cost functions
Regular cost functions

Regular languages:
  toolbox for solving boolean problems over words and trees

Regular cost functions generalize it to:
  toolbox for solving boundedness questions over words and trees
Regular cost functions

Regular languages:
  toolbox for solving boolean problems over words and trees
Regular cost functions generalize it to:
  toolbox for solving boundedness questions over words and trees

For instance, can you bound $n$ such that:

$$(L + \varepsilon)^n = L^* \quad ?$$

(Finite power property [Simon])
Regular cost functions

Regular languages:

  toolbox for solving boolean problems over words and trees

Regular cost functions generalize it to:

  toolbox for solving boundedness questions over words and trees

For instance, can you bound $n$ such that:

$$(L + \varepsilon)^n = L^* \ ?$$

(Finite power property [Simon])

Over all words $u$ (infinite trees $t$), the fixpoint of $\phi(x,Z)$ is reached within at most $n$ steps?

(boundedness of fixpoint [Blumensath&Otto&Weyer])
Regular cost functions

Regular languages:
  toolbox for solving boolean problems over words and trees
Regular cost functions generalize it to:
  toolbox for solving boundedness questions over words and trees

For instance, can you bound $n$ such that:

$$(L + \varepsilon)^n = L^*$$

Over all words $u$ (infinite trees $t$), the fixpoint of $\phi(x,Z)$ is reached within at most $n$ steps?

(Finite power property [Simon])

(boundedness of fixpoint [Blumensath&Otto&Weyer])

For all words $u$ (tree $t$), there exists a regular expression of star-height $k$ of size $n$ that accepts a subset of $L$ that contains $u$ (resp. $t$)?

(star-height problem [Hashiguchi, Kirsten, C&Löding])
Regular cost functions

Regular languages:
  toolbox for solving boolean problems over words and trees
Regular cost functions generalize it to:
  toolbox for solving boundedness questions over words and trees

For instance, can you bound $n$ such that:

$$(L + \epsilon)^n = L^*$$  \quad \text{(Finite power property [Simon])}

Over all words $u$ (infinite trees $t$), the fixpoint of $\phi(x,Z)$ is reached within at most $n$ steps?  \quad \text{(boundedness of fixpoint [Blumensath&Otto&Weyer])}

For all words $u$ (tree $t$), there exists a regular expression of star-height $k$ of size $n$ that accepts a subset of $L$ that contains $u$ (resp. $t$)?  \quad \text{(star-height problem [Hashiguchi, Kirsten, C&Löding])}

For all infinite trees $t$, there exists a parity automaton of index $[i,j]$, and size $n$ that accepts a subset of $L$ that contains $t$?  \quad \text{(Mostowski [CL,CKLvB])}
Regular cost functions

Regular languages:
    toolbox for solving boolean problems over words and trees

Regular cost functions generalize it to:
    toolbox for solving boundedness questions over words and trees

For instance, can you bound n such that:

\[(L + \epsilon)^n = L^* \quad ?\]  
(Finite power property [Simon])

Over all words u (infinite trees t), the fixpoint of \(\phi(x,Z)\) is reached within at most n steps?  
(boundedness of fixpoint [Blumensath&Otto&Weyer])

For all words u (tree t), there exists a regular expression of star-height k of size n that accepts a subset of L that contains u (resp. t)?  
(star-height problem [Hashiguchi, Kirsten, C&Löding])

For all infinite trees t, there exists a parity automaton of index [i,j], and size n that accepts a subset of L that contains t?  
(Mostowski [CL,CKLvB])

The Church synthesis problem can be solved up to an error of n bits?  
([Rabinovitch&Velner])
Regular cost functions

Regular languages:
  toolbox for solving boolean problems over words and trees
Regular cost functions generalize it to:
  toolbox for solving boundedness questions over words and trees

For instance, can you bound $n$ such that:

$$(L + \varepsilon)^n = L^*$$

(Finite power property \[Simon\])

Over all words $u$ (infinite trees $t$), the fixpoint of $\phi(x,Z)$ is reached within at most $n$ steps?
(boundedness of fixpoint \[Blumensath&Otto&Weyer\])

For all words $u$ (tree $t$), there exists a regular expression of star-height $k$ of size $n$ that accepts a subset of $L$ that contains $u$ (resp. $t$)?
(star-height problem \[Hashiguchi, Kirsten, C&Löding\])

For all infinite trees $t$, there exists a parity automaton of index $[i,j]$, and size $n$ that accepts a subset of $L$ that contains $t$?  (Mostowski \[CL,CKLvB\])

The Church synthesis problem can be solved up to an error of $n$ bits?  (\[Rabinovitch&Velner\])

For all these questions, we do not care about precise values.
Regular cost functions ideas
Regular cost functions ideas

A regular cost function $f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$
can be seen as two (unusual) languages:
A regular cost function $f : A^* \to \mathbb{N} \cup \{\infty\}$ can be seen as two (unusual) languages:

$$\{ u : f(u) \text{ is small} \}$$
Regular cost functions ideas

A regular cost function $f : A^* \to \mathbb{N} \cup \{\infty\}$ can be seen as two (unusual) languages:

\[
\{ u : f(u) \text{ is small} \} \quad \{ u : f(u) \text{ is large} \}
\]
A regular cost function $f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ can be seen as two (unusual) languages:

\{ u : f(u) is small \} \quad \text{complement} \quad \{ u : f(u) is large \}
A regular cost function \( f : A^* \rightarrow \mathbb{N} \cup \{\infty\} \) can be seen as two (unusual) languages:

\[ \{ u : f(u) \text{ is small} \} \quad \text{complement} \quad \{ u : f(u) \text{ is large} \} \]

It has a formal meaning in non-standard analysis.
Regular cost functions ideas

A regular cost function \( f : A^* \rightarrow \mathbb{N} \cup \{\infty\} \) can be seen as two (unusual) languages:

- \( \{ u : f(u) \text{ is small} \} \)
- \( \{ u : f(u) \text{ is large} \} \)

It has a formal meaning in non-standard analysis.

Functions that are large on the same inputs are \( \approx \)-equivalent.
Regular cost functions ideas

A regular cost function \( f : A^* \rightarrow \mathbb{N} \cup \{\infty\} \) can be seen as two (unusual) languages:

\[
\{ u : f(u) \text{ is small} \} \quad \text{ext} \quad \text{complement} \quad \{ u : f(u) \text{ is large} \} \quad \text{ext}
\]

Functions that are large on the same inputs are \( \approx \)-equivalent.

Example: \( f(u) = \) the length of longest block of consecutive a’s

\[
(a^{\text{small}} b)^* a^{\text{small}} \quad \text{complement} \quad (a^* b)^* a^{\text{large}} (b a^*)^*
\]

It has a formal meaning in non-standard analysis.
A regular cost function $f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ can be seen as two (unusual) languages:

$$\{ u : f(u) \text{ is small} \}$$

and

$$\{ u : f(u) \text{ is large} \}$$

Functions that are large on the same inputs are $\approx$-equivalent.

Example: $f(u) = \text{the length of longest block of consecutive a's}$

$$\left(a^{\text{small}}b\right)^* a^{\text{small}}$$

and

$$\left(a^{\text{large}}b\right)^* a^{\text{large}} (ba^*)^*$$

Main theorem of regular cost functions:

- B-rational expressions
- B-automata
- cost MSO
- (stabilisation monoids, up-sets)

- S-rational expressions
- S-automata
- cost MSO*
- (stabilisation monoids, down-sets)

It has a formal meaning in non-standard analysis.
Difficulties of REG (cost functions)
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly.
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly.

Constructions are complicated, and in particular more complicated than for infinite words.
Difficulties of \text{REG}(\text{cost functions})

Manipulating bounds everywhere in the proofs is costly.

Constructions are \textbf{complicated}, and in particular more complicated than for infinite words.

Automata \textbf{cannot be determinized}, while it is required for treating the case of trees.

One has to resort in \textbf{history-deterministic} automata which are more involved to handle.
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly. Constructions are complicated, and in particular more complicated than for infinite words.

Automata cannot be determinized, while it is required for treating the case of trees. One has to resort in history-deterministic automata which are more involved to handle.

Similarities with REG(infinite objects)
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly.

Constructions are complicated, and in particular more complicated than for infinite words.

Automata cannot be determinized, while it is required for treating the case of trees.

One has to resort in history-deterministic automata which are more involved to handle.

Similarities with REG(infinite objects)

\[(ab)^n a = a(ba)^n \quad (n \text{ large})\]

\[(ab)^\omega = a(ba)^\omega\]
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly.

Constructions are **complicated**, and in particular more complicated than for infinite words.

Automata **cannot be determinized**, while it is required for treating the case of trees.

One has to resort in **history-deterministic** automata which are more involved to handle.

**Similarities with REG(infinite objects)**

\[(ab)^n a = a(ba)^n \quad (n\text{ large})\]

\[(ab)^\omega = a(ba)^\omega\]

High levels ideas are similar:
- use of games for dealing with trees
- use of the ideal decomposition of monoids
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly.

Constructions are **complicated**, and in particular more complicated than for infinite words.

Automata **cannot be determinized**, while it is required for treating the case of trees.

One has to resort in **history-deterministic** automata which are more involved to handle.

**Similarities with REG(infinite objects)**

\[(ab)^n a = a(ba)^n \quad (n \text{ large})\]
\[(ab)^\omega = a(ba)^\omega\]

High levels ideas are similar:
- use of games for dealing with trees
- use of the ideal decomposition of monoids

The most efficient translations from B/S-automata to history-deterministic B/S-automata mimic the ideas of **Safra's construction**.
Difficulties of REG(cost functions)

Manipulating bounds everywhere in the proofs is costly.

Constructions are complicated, and in particular more complicated than for infinite words.

Automata cannot be determinized, while it is required for treating the case of trees.
One has to resort in history-deterministic automata which are more involved to handle.

Similarities with REG(infinite objects)

\[(ab)^n a = a(ba)^n \quad (n \text{ large})\]
\[(ab)^\omega = a(ba)^\omega\]

High levels ideas are similar:
- use of games for dealing with trees
- use of the ideal decomposition of monoids

The most efficient translations from B/S-automata to history-deterministic B/S-automata mimic the ideas of Safra's construction.

This work makes formal some similarities, and use it to factorizing proofs.
The bridge
The bridge

$L \subseteq A^\omega$

Regular languages of \(\omega\)-words
$L \subseteq A^\omega$

Regular languages of $\omega$-words

$f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$

(up to $\approx$)

Regular cost functions
The bridge

Definition: For \( L \subseteq A^\omega \),

\[
L^{\omega 1} : A^* \rightarrow \mathbb{N} \cup \{\infty\}
\]

\[
u \mapsto \sup\{n : u = vw_1 \ldots w_n v' \mid w_1, \ldots, w_n \neq \varepsilon, v\{w_1, \ldots, w_n\}^\omega \subseteq L\}
\]
The bridge

\[ L \subseteq A^\omega \]

\textbf{Definition:} For \( L \subseteq A^\omega \),

\[ L^{o1} : A^* \rightarrow \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_n v' \}
\]

\[ w_1, \ldots, w_n \neq \varepsilon \]

\[ v\{w_1, \ldots, w_n\}^\omega \subseteq L \]

\[ f : A^* \rightarrow \mathbb{N} \cup \{\infty\} \text{ (up to } \approx) \]

\textbf{Lemma:} For \( L \subseteq A^\omega \) regular, \( L^{o1} \) is a regular cost function.
**The bridge**

\[ L \subseteq A^\omega \]

Regular languages of \( \omega \)-words

\[ L^0_1 : A^* \to \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_n v' \mid w_1, \ldots, w_n \neq \varepsilon, v\{w_1, \ldots, w_n\}^\omega \subseteq L\} \]

**Definition:** For \( L \subseteq A^\omega \),

\[ L^0_1 : A^* \to \mathbb{N} \cup \{\infty\} \]

**Lemma:** For \( L \subseteq A^\omega \) regular, \( L^0_1 \) is a regular cost function.

\[ f : A^* \to \mathbb{N} \cup \{\infty\} \]

(up to \( \approx \))

Regular cost functions
The bridge $f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ (up to $\approx$)

**Definition:** For $L \subseteq A^\omega$, $L^{o1} : A^* \rightarrow \mathbb{N} \cup \{\infty\}$

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \mid w_1, \ldots, w_n \neq \varepsilon, v\{w_1, \ldots, w_n\}^\omega \subseteq L\} \]

**Lemma:** For $L \subseteq A^\omega$ regular, $L^{o1}$ is a regular cost function.

**Lemma:** $uv^\omega \in L$ iff 

\[ \lim_{n} L^{o1}(uv^n) = \infty \]
The bridge $f : A^* \to \mathbb{N} \cup \{\infty\}$ (up to $\approx$)

**Regular languages of $\omega$-words**

**Definition:** For $L \subseteq A^\omega$, let $L^{o1} : A^* \to \mathbb{N} \cup \{\infty\}$

$$u \mapsto \sup\{n \mid u = vw_1 \ldots w_nv' \quad \text{such that} \quad w_1, \ldots, w_n \neq \varepsilon \quad \text{and} \quad v\{w_1, \ldots, w_n\}^\omega \subseteq L\}$$

**Lemma:** For $L \subseteq A^\omega$ regular, $L^{o1}$ is a regular cost function.

**Lemma:** $uv^\omega \in L$ iff

$$\lim_{n \to \infty} L^{o1}(uv^n) = \infty$$

**Lemma:** The $^{o1}$ map is injective.
The bridge

\[ f : A^* \rightarrow \mathbb{N} \cup \{\infty\} \] (up to \(\approx\))

\[ L \subseteq A^\omega \]

Regular cost functions

Regular languages of \(\omega\)-words

\(L^\omega : A^* \rightarrow \mathbb{N} \cup \{\infty\}\)

\[
u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \mid w_1, \ldots, w_n \neq \epsilon \}
\]

\[
v\{w_1, \ldots, w_n\}^\omega \subseteq L\]

\(L\omega^1\) is a regular cost function.

Lemma: For \(L \subseteq A^\omega\) regular, \(L\omega^1\) is a regular cost function.

Lemma: \(uv^\omega \in L\) iff \[\lim_{n} L\omega^1(uvn) = \infty\]

Lemma: The \(\omega^1\) map is injective.
First examples of the bridge

$L \subseteq A^\omega$  $\rightarrow$  

$L^{o1} : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ 

$u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \}$

$w_1, \ldots, w_n \neq \varepsilon$

$u\{w_1, \ldots, w_n\}^\omega \subseteq L$
First examples of the bridge

\[ L \subseteq A^\omega \quad \xrightarrow{\quad} \quad L^01 : A^* \rightarrow \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \]

\[ w_1, \ldots, w_n \neq \varepsilon \]

\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \]

\[ Buchi = (1*2)^\omega \]
First examples of the bridge

$L \subseteq A^\omega$  \hspace{2cm}  $L^{o1} : A^* \rightarrow \mathbb{N} \cup \{\infty\}$

$u \mapsto \sup\{n : u = vw_1 \ldots w_n v' \}
\quad w_1, \ldots, w_n \neq \varepsilon
\quad u\{w_1, \ldots, w_n\}^\omega \subseteq L$}

$Buchi = (1^*2)^\omega$  \hspace{2cm}  $| \cdot |_2$
First examples of the bridge

\[ L \subseteq A^\omega \]

\[ L^{01} : A^* \rightarrow \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_n v' \] \[ w_1, \ldots, w_n \neq \varepsilon \]

\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \] 

\[ Buchi = (1^*2)^\omega \]

\[ coBuchi = (0 + 1)^*0^\omega \]
First examples of the bridge

\[ L \subseteq A^\omega \]

\[ L^{o1} : A^* \to \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n \; : \; u = vw_1 \ldots w_n v' \quad \]

\[ w_1, \ldots, w_n \neq \varepsilon \]

\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \}

\[ Buchi = (1*2)^\omega \]

\[ coBuchi = (0 + 1)^* 0^\omega \]

\[ | \cdot |_2 \]

\[ maxblock_0 \]
First examples of the bridge

\[ L \subseteq A^\omega \]

\[ L^{01} : A^* \rightarrow \mathbb{N} \cup \{\infty\} \]
\[ u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \}
\[ w_1, \ldots, w_n \neq \varepsilon \]
\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \]

\[ Buchi = (1^*2)^\omega \]

\[ coBuchi = (0 + 1)^*0^\omega \]

\[ parity_{i,j} = \{u \in [i, j]^\omega : \limsup_{n} u_n \text{ even}\} \]

\[ | \cdot |_2 \]

\[ maxblock_0 \]
First examples of the bridge

$$L \subseteq A^\omega$$

$$L^{01} : A^* \rightarrow \mathbb{N} \cup \{\infty\}$$

$$u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \mid w_1, \ldots, w_n \neq \varepsilon \}$$

$$u\{w_1, \ldots, w_n\}^\omega \subseteq L \}$$

$$Buchi = (1^*2)^\omega$$

$$coBuchi = (0 + 1)^*0^\omega$$

$$parity_{i,j} = \{u \in [i, j]^\omega \mid \limsup_{n} u_n \text{ even}\}$$

$$maxblock_0$$

hierarchical B-condition
First examples of the bridge

\[ L \subseteq A^\omega \quad \text{↔} \quad L^{\text{co1}} : A^* \to \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_n v' \}
\]

\[ w_1, \ldots, w_n \neq \varepsilon \]

\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \]

\[ \text{Buchi} = (1^*2)^\omega \]

\[ \text{coBuchi} = (0 + 1)^*0^\omega \]

\[ \text{parity}_{i,j} = \{u \in [i,j]^\omega \mid \limsup_{n} u_n \text{ even}\} \]

\[ \text{maxblock}_0 \]

\[ \| \cdot \|_2 \]

\[ \text{hierarchical B-condition (up to } \approx) \]
First examples of the bridge

$L \subseteq A^\omega \quad \rightarrow \quad L^{\ominus 1} : A^* \rightarrow \mathbb{N} \cup \{\infty\}$

\[ L^{\ominus 1} : A^* \rightarrow \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n \mid u = vw_1 \ldots w_n v' \quad w_1, \ldots, w_n \neq \varepsilon \quad u\{w_1, \ldots, w_n\}^\omega \subseteq L\} \]

\[ Buchi = (1*2)^\omega \]

\[ coBuchi = (0 + 1)^*0^\omega \]

\[ parity_{i,j} = \{u \in [i,j]^\omega \mid \limsup_n u_n \text{ even}\} \quad \text{hierarchical B-condition (up to } \approx) \]

\[ Büchi \text{ automaton} \]

\[ \| \cdot \|_2 \quad maxblock_0 \]
First examples of the bridge

\[ L \subseteq A^\omega \]

\[ L^{o1} : A^* \to \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup\{n : u = vw_1 \ldots w_nv' \}
\]

\[ w_1, \ldots, w_n \neq \varepsilon \]

\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \]

\[ Buchi = (1^*2)^\omega \]

\[ coBuchi = (0 + 1)^*0^\omega \]

\[ \text{parity}_{i,j} = \{u \in [i, j]^\omega | \limsup_n u_n \text{ even}\} \]

\[ \text{Buchi automaton} \]

\[ \lVert \cdot \rVert_2 \]

\[ \text{maxblock}_0 \]

\[ \text{hierarchical B-condition} \]

\[ (\text{up to } \approx) \]

\[ \text{Same automaton seen as} \]

\[ \text{max-prefix-distance} \]

\[ (\text{up to } \approx) \]
Büchi automaton case
A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.
A Büchi automaton accepts the language \( L \) of \( \omega \)-words such that there is a (infinite) run with infinitely many \textit{Büchi transitions}. Seen as a \textit{max-prefix-distance}, it computes for a finite \( u \):

\[
f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.
\]
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Seen as a max-prefix-distance, it computes for a finite $u$: $f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u$.

A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.
A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: $L^{\omega_1} \approx f$

Seen as a max-prefix-distance, it computes for a finite $u$: $f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u$.

A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: $L^{o1} \simeq f$

I.e. $L^{o1}(u)$ large iff $f(u)$ large.

Seen as a max-prefix-distance, it computes for a finite $u$: $f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u$. A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: \( L^{o1} \approx f \)

I.e. \( L^{o1}(u) \) large iff \( f(u) \) large.

Assume \( f(u) \) large.

Seen as a max-prefix-distance, it computes for a finite \( u \):
\[ f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u. \]

A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

**Goal:** $L^{o_1} \approx f$

I.e. $L^{o_1}(u)$ large iff $f(u)$ large.

Assume $f(u)$ large.

There is a prefix-run over $u$ with a large number of Büchi transitions.

Seen as a **max-prefix-distance**, it computes for a finite $u$: $f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u$.

A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.
Büchi automaton case

A Büchi automaton accepts the language \( L \) of \( \omega \)-words such that there is a (infinite) run with infinitely many Büchi transitions.

Seen as a max-prefix-distance, it computes for a finite \( u \):
\[
f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.
\]

Goal: \( L^{o1} \simeq f \)

I.e. \( L^{o1}(u) \) large iff \( f(u) \) large.

Assume \( f(u) \) large.

There is a prefix-run over \( u \) with a large number of Büchi transitions.

It can be decomposed into \( \alpha \beta_1 \ldots \beta_n \gamma \) with \( n \) large and each of the \( \beta \)'s start and end in the state state, and contain at least one Büchi transition.

A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Seen as a max-prefix-distance, it computes for a finite $u$: $f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u$.

Goal: $L^{o1} \approx f$

I.e. $L^{o1}(u)$ large iff $f(u)$ large.

Assume $f(u)$ large.

There is a prefix-run over $u$ with a large number of Büchi transitions.

It can be decomposed into $\alpha\beta_1 \ldots \beta_n\gamma$ with $n$ large and each of the $\beta$'s start and end in the state state, and contain at least one Büchi transition.

Hence $\alpha\{\beta_1, \ldots, \beta_n\}^\omega$ is a set of accepting runs.
Büchi automaton case

A Büchi automaton accepts the language \( L \) of \( \omega \)-words such that there is a (infinite) run with infinitely many Büchi transitions.

Seen as a max-prefix-distance, it computes for a finite \( u \):
\[ f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u. \]

A sequence of transitions starting in an initial state, forming a path reading the prefix of the input.

Goal: \( L^{01} \approx f \)

I.e. \( L^{01}(u) \) large iff \( f(u) \) large.

Assume \( f(u) \) large.

There is a prefix-run over \( u \) with a large number of Büchi transitions.

It can be decomposed into \( \alpha \beta_1 \ldots \beta_n \gamma \) with \( n \) large and each of the \( \beta \)'s start and end in the state state, and contain at least one Büchi transition.

Hence \( \alpha \{ \beta_1, \ldots, \beta_n \}^\omega \) is a set of accepting runs.

Hence \( L^{01}(u) \geq n \) is large.
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Seen as a max-prefix-distance, it computes:

$$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.$$ 

Goal: $L^{o1} \simeq f$
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: $L^{o1} \simeq f$

Assume $L^{o1}(u)$ is large.

Seen as a max-prefix-distance, it computes:

$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.$
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: $L^{o_1} \approx f$

Assume $L^{o_1}(u)$ is large.

$u = v \ w_1 \ w_2 \ldots \ w_n \ t$

with $n$ large,

and $v\{w_1, \ldots, w_n\}^\omega \subseteq L$

Seen as a max-prefix-distance, it computes:

$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u$. 
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: $L^{\infty} \approx f$

Assume $L^{\infty}(u)$ is large.

$u = v w_1 w_2 \ldots w_n t$

By Ramsey these can be regrouped into:

Seen as a max-prefix-distance, it computes:

$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.$

with $n$ large, and $v\{w_1, \ldots, w_n\}^\omega \subseteq L$
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Goal: $L^{o1} \approx f$

Assume $L^{o1}(u)$ is large.

$$u = v \, w_1 \, w_2 \, \ldots \, w_n \, t$$

By Ramsey these can be regrouped into:

$$u = v' \, w'_1 \, w'_2 \, \ldots \, w'_n \, t'$$

Seen as a max-prefix-distance, it computes:

$$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.$$ 

with $n$ large, and

$$v\{w_1, \ldots, w_n\}^\omega \subseteq L$$

with $n'$ large, all $w''$'s corresponding to the same idempotent $e$ in the transition semigroup of the automaton:

$$\mathcal{P}(Q \times \{1, 2\} \times Q)$$
A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

**Goal:** $L^{o1} \approx f$

Assume $L^{o1}(u)$ is large.

$$u = v \, w_1 \, w_2 \, \ldots \, w_n \, t$$

By Ramsey these can be regrouped into:

$$u = v' \, w'_1 \, w'_2 \, \ldots \, w'_n \, t'$$

Since $v'\{w'_1, \ldots, w'_n\}^{\omega} \subseteq L$

this idempotent contains some `reachable' $(p,2,p)$.

Seen as a max-prefix-distance, it computes:

$$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.$$
Büchi automaton case

A Büchi automaton accepts the language $L$ of $\omega$-words such that there is a (infinite) run with infinitely many Büchi transitions.

Seen as a max-prefix-distance, it computes:

$$f(u) = \text{maximum number of Büchi transitions seen on a prefix-run over } u.$$ 

Goal: $L^o \approx f$

Assume $L^o(u)$ is large.

$$u = v \ w_1 \ w_2 \ \ldots \ w_n \ t$$

By Ramsey these can be regrouped into:

$$u = v' \ w'_1 \ w'_2 \ \ldots \ w'_n \ t'$$

Since $v'\{w'_1, \ldots, w'_n\}^\omega \subseteq L$

this idempotent contains some `reachable' $(p,2,p)$.

This witnesses that $f(u)$ is large.
First examples of the bridge

$L \subseteq A^\omega$ \quad $L^{o1} : A^* \to \mathbb{N} \cup \{\infty\}$

\[ u \mapsto \sup \{n : u = vw_1 \ldots w_nv' \quad w_1, \ldots, w_n \neq \varepsilon \quad u\{w_1, \ldots, w_n\}^\omega \subseteq L \} \]

$\text{Buchi} = (1^*2)^\omega$

$\text{coBuchi} = (0 + 1)^*0^\omega$

$\text{parity}_{i,j} = \{u \in [i, j]^\omega | \limsup_n u_n \text{ even} \}$

Büchi automaton \quad Same automaton seen as max-prefix-distance (up to $\approx$)

hierarchical B-condition (up to $\approx$)

$| \cdot |_2$ \quad $\text{maxblock}_0$
First examples of the bridge

\[ L \subseteq A^\omega \quad \quad L^{o1} : A^* \rightarrow \mathbb{N} \cup \{\infty\} \]

\[ u \mapsto \sup \{ n : u = vw_1 \ldots w_n v' \}
\]

\[ w_1, \ldots, w_n \neq \varepsilon \]

\[ u\{w_1, \ldots, w_n\}^\omega \subseteq L \]

\[ \text{Buchi} = (1*2)^\omega \quad \quad | \cdot |_2 \]

\[ \text{coBuchi} = (0 + 1)^*0^\omega \quad \quad \text{maxblock}_0 \]

\[ \text{parity}_{i,j} = \{ u \in [i,j]^\omega \mid \limsup_n u_n \text{ even} \} \quad \quad \text{hierarchical B-condition} \quad \quad \text{(up to } \approx) \]

Büchi automaton

Rabin automaton

Same automaton seen as
max-prefix-distance (up to \( \approx \))

Same automaton seen as
max-prefix-B automaton (up to \( \approx \))
The bridge

Regular languages of $\omega$-words

$\omega$-regular like cost functions

Regular cost functions
Consequence: For all $\omega$-regular like cost function, there exists effectively a $B$-deterministic $a$ that recognizes it (up to $\approx$).
The bridge

Regular languages of \( \omega \)-words

\( \omega \)-regular like cost functions

\[ L^{\circ_1} \]

\( L \)

**Consequence:** For all \( \omega \)-regular like cost function, there exists effectively a B-deterministic \( a \) that recognizes it (up to \( \approx \)).

**Proof:** An \( \omega \)-regular like cost function is of the form \( L^{\circ_1} \) for some regular language of \( \omega \)-words \( L \).
The bridge

**Consequence:** For all $\omega$-regular like cost function, there exists effectively a B-deterministic $a$ that recognizes it (up to $\approx$).

**Proof:** An $\omega$-regular like cost function is of the form $L^{o1}$ for some regular language of $\omega$-words $L$.

There exists a deterministic Rabin automaton for it [McNaughton/Safra].
**The bridge**

**Consequence:** For all $\omega$-regular like cost function, there exists effectively a B-deterministic $a$ that recognizes it (up to $\approx$).

**Proof:** An $\omega$-regular like cost function is of the form $L^o$ for some regular language of $\omega$-words $L$.

There exists a deterministic Rabin automaton for it [McNaughton/Safra]. This is a max-prefix-B-automaton for $L^o$. 
**Consequence:** For all ω-regular like cost function, there exists effectively a B-deterministic a that recognizes it (up to ≈).

**Proof:** An ω-regular like cost function is of the form $L^{o1}$ for some regular language of ω-words $L$.

There exists a deterministic Rabin automaton for it [McNaughton/Safra]. This is a max-prefix-B-automaton for $L^{o1}$.

Since it is deterministic and complete it is in fact a deterministic-B-automaton (all states set to final).
**Consequence:** For all $\omega$-regular like cost function, there exists effectively a B-deterministic $A$ that recognizes it (up to $\approx$).

**Proof:** An $\omega$-regular like cost function is of the form $L^{o1}$ for some regular language of $\omega$-words $L$.

There exists a deterministic Rabin automaton for it [McNaughton/Safra]. This is a max-prefix-B-automaton for $L^{o1}$.

Since it is deterministic and complete it is in fact a deterministic-B-automaton (all states set to final).
Consequence: For all $\omega$-regular like cost function, there exists effectively a $B$-deterministic $a$ that recognizes it (up to $\approx$).

Proof: An $\omega$-regular like cost function is of the form $L^{o1}$ for some regular language of $\omega$-words $L$.

There exists a deterministic Rabin automaton for it [McNaughton/Safra]. This is a max-prefix-$B$-automaton for $L^{o1}$.

Since it is deterministic and complete it is in fact a deterministic-$B$-automaton (all states set to final).
Why do we care?
Why do we care?

I am interested in regular cost functions, not the special case of \( \omega \)-regular like cost functions. So why do I care if this subclass inherits all the good properties and constructions of the regular languages of \( \omega \)-words?
What is history-determinism?

good-for-games automata [Henzinger&Piterman]
What is history-determinism?

(good-for-games automata [Henzinger&Piterman])

An **history deterministic automaton** is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.
What is history-determinism?

(\textit{good-for-games automata \cite{Henzinger&Piterman}})

An \textbf{history deterministic automaton} is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.

\textbf{Example one-counter-B:} A counter can be incremented or reset. Maximal value counts. Infimum over all runs.
What is history-determinism?
(good-for-games automata [Henzinger&Piterman])

An **history deterministic automaton** is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.

**Example one-counter-B:** A counter can be incremented or reset. Maximal value counts. Infimum over all runs.
An **history deterministic automaton** is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.

**Example one-counter-B:** A counter can be incremented or reset. Maximal value counts. Infimum over all runs.

\[
\minblock(a^{n_0}ba^{n_1} \ldots ba^{n_k}) = \min(n_0, \ldots, n_k)
\]
What is history-determinism? (good-for-games automata [Henzinger&Piterman])

An **history deterministic automaton** is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.

**Example one-counter-B:** A counter can be incremented or reset. Maximal value counts. Infimum over all runs.

\[ \minblock(a^{n_0}ba^{n_1} \ldots ba^{n_k}) = \min(n_0, \ldots, n_k) \]
What is history-determinism?

(Anatomy-for-games automata [Henzinger&Piterman])

An history deterministic automaton is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.

Example one-counter-B: A counter can be incremented or reset. Maximal value counts. Infimum over all runs.

\[
\text{minblock}(a^{n_0}b a^{n_1} \ldots b a^{n_k}) = \min(n_0, \ldots, n_k)
\]

These automata are semantically deterministic, but not syntactically.
What is history-determinism?

(An history deterministic automaton is a non-deterministic automaton such that an almighty oracle can decide what is the best transition to take knowing the run so far.

Example one-counter-B: A counter can be incremented or reset. Maximal value counts. Infimum over all runs.

\[
\minblock(a^{n_0}ba^{n_1} \ldots ba^{n_k}) = \min(n_0, \ldots, n_k)
\]

These automata are semantically deterministic, but not syntactically. These are as good as deterministic automata when run in a branching context (i.e. a tree or a game).
Why do we care?

I am interested in regular cost functions, not the special case of \( \omega \)-regular like cost functions. So why do I care if this subclass inherits all the good properties and constructions of the regular languages of \( \omega \)-words?
Why do we care?

I am interested in regular cost functions, not the special case of \( \omega \)-regular like cost functions. So why do I care if this subclass inherits all the good properties and constructions of the regular languages of \( \omega \)-words?

Because (inspired from [Bojanczyk15]):

\[
\text{determinization of } \omega \text{-regular like cost functions} + \text{positional determinacy of hierarchical B-games [C&Löding06]} = \text{history-determinization of regular cost functions [C09,C11 unp]}
\]
Conclusion

We have provided a bridge from regular languages of infinite words to a subset of regular cost functions over finite words.

This allows to transport constructions and results from the well studied and simpler theory of regular languages of infinite words to cost functions over infinite trees.

In particular a new, simple and optimal proof for transforming B-automata in historic-deterministic form can be derived (a central result for working on games and trees).
Conclusion

We have provided a bridge from regular languages of infinite words to a subset of regular cost functions over finite words.

This allows to transport constructions and results from the well studied and simpler theory of regular languages of infinite words to cost functions over infinite trees.

In particular a new, simple and optimal proof for transforming B-automata in historic-deterministic form can be derived (a central result for working on games and trees).

TODO:
Extend the approach to produce history deterministic S-automata.